Greedy Flipping for Constrained Word Deletion

Jin-ge Yao, Xiaojun Wan
Institute of Computer Science and Technology, Peking University, Beijing 100871, China
The MOE Key Laboratory of Computational Linguistics, Peking University
{yaojinge, wanxiaojun}@pku.edu.cn

Abstract

In this paper we propose a simple yet efficient method for constrained word deletion to compress sentences, based on top-down greedy local flipping from multiple random initializations. The algorithm naturally integrates various grammatical constraints in the compression process, without using time-consuming integer linear programming solvers. Our formulation suits for any objective function involving arbitrary local score definition. Experimental results show that the proposed method achieves nearly identical performance with explicit ILP formulation while being much more efficient.

Introduction

Sentence compression is the task of generating a grammatical and usually shorter summary for a long sentence, while preserving its most important information. This task has aroused much attention because of its potential applications, including the generation of headlines and subtitles, text display on small screens and non-extractive summarization. One specific instantiation of sentence compression is word deletion, namely generating a compression by dropping words from the original sentence. This is also known as *extractive* compression, as opposed to *abstractive* compression which allows more elaborate transformations other than word deletion such as lexical substitution.

Although some related studies start to transfer focus onto abstractive methods (Woodsend and Lapata 2011; Narayan and Gardent 2014; Rush, Chopra, and Weston 2015; Xu et al. 2016), deletion-based compression is still far from being solved. Meanwhile, it has been shown that extractive methods can still become more suitable than abstractive methods in many scenarios (Nomoto 2009).

Various approaches have been proposed to challenge the task of deletion-based compression. Earlier pioneering studies (Knight and Marcu 2000; Turner and Charniak 2005) consider several insightful approaches, including generative noisy-channel models and discriminative decision tree models. Structured discriminative compression models (McDonald 2006) are capable of integrating rich features and have been shown effective. More recently, sequence-to-sequence learning with gated recurrent nets has also been applied for deletion-based compression (Filippova et al. 2015), utilizing large-scale parallel training data. A potential weakness for these approaches is the lack of explicit reassurance for the generated compressions to be syntactically grammatical.

To ensure that the generated compressions remain to be grammatical, there are two major paradigms. The first is explicitly rewrite sentences with pre-defined synchronous grammars based on constituent parse trees, known as tree transduction (Cohn and Lapata 2007; 2009; Yamangil and Shieber 2010). In such models a synchronous grammar is induced from a corpus of parallel constituent trees. Compressions are generated from the grammar to maximize some weighted scores.

The other paradigm is constraints-based models, i.e. explicitly impose syntactic constraints to make the final compressions grammatical. Many sentence compression models that explicitly deal with syntactic constraints are mostly based on integer linear programming (ILP) (Clarke and Lapata 2008; Filippova and Strube 2008; Thadani and McKeown 2013) or Markov logic networks (MLN) (Huang et al. 2012; Yoshikawa et al. 2012). The constraints are mostly derived from dependency parsing, which is usually much cheaper in terms of computation, compared with constituent parsing. Unfortunately inference problems in ILP models and MLNs are in general NP-hard, which limits the efficiency and scalability of such approaches.

To enhance local smoothness in practice, language models are usually introduced when decoding tree transduction models and ILP models. This brings the challenge that the scoring function now consists of different kinds of local structures (ngrams in LM and local tree structure), eliminating the possibility for efficient decoding using dynamic programming and making ILP models even slower in practice. One may use Lagrangian relaxation and dual decomposition for approximate inference (Thadani 2014; Yao, Wan, and Xiao 2014) for structured models, utilizing the fact that the score function can be decomposed into subparts for efficient independent decoding followed by merging the solutions from the subparts.

In this paper we present a much simpler yet more efficient solution for sentence compression, enabling: (1) syntactic constraints ensuring grammaticality, and (2) rich scoring functions containing different kinds of locality or structures. Starting with an initial random bit vector indicating binary decisions for word deletion, the algorithm traverses the dependency tree and locally flips one (or more, according to certain constraints) bit(s) as an attempt to improve the scoring function, injecting local grammatical constraints during this process. Multiple random initializations will be conducted to tackle the problem of local optimality. Experiments show that the proposed simple algorithm can yield results that are close to exact ILP formulations, while being much more efficient in terms of time cost.

Constrained Word Deletion

The Problem

Sentence compression is usually expressed as a combinatorial optimization problem. Given a long sentence $\mathbf{x} = \{x_1, \ldots, x_n\}$ to be compressed, ILP-based models encode the decision to keep or delete each word x_i as an indicator variable $\delta_i \in \{0, 1\}$ and we denote word deletions as setting corresponding δ_i to be 0. Then any candidate compression can be expressed as a bit vector $\mathbf{d} = \{\delta_1, \ldots, \delta_n\}$. The objective function $F_{\mathbf{x}}(\mathbf{d})$ is in the form of:

$$F_{\mathbf{x}}(\mathbf{d}) = S(\mathbf{x}, \mathbf{d}) + \sum_{i=1}^{n} \delta_i \cdot w(x_i) \tag{1}$$

where $S(\mathbf{x}, \mathbf{d})$ is a rich scoring function for a compression $\mathbf{d}$ on $\mathbf{x}$, defined on local structures such as bigrams and trigrams, while $w(x_i)$ is a weighing function for word x_i. For dependency-based compression (Filippova and Strube 2008) we can also view $w(x_i)$ in (1) as the score for a dependency arc with the word x_i as modifier, while δ_i being the indicator for keeping/deleting that arc. From either view, the objective function $F_{\mathbf{x}}(\mathbf{d})$ consists of different types of locality, making it difficult to optimize efficiently.

One solution is to cast the problem of maximizing (1) as integer linear programming (ILP) (Clarke and Lapata 2008). More specifically, we can introduce a variable to denote the binary decisions on each local structure (e.g. a trigram), with a bunch of additional constraints to let them being consistent with the bit vector $\mathbf{d} = \{\delta_1, \ldots, \delta_n\}$. [1] This will lead to a significant growth of complexity for optimization. For example, trigram-based scoring function will introduce additional $O(n^3)$ variables and a bunch of coherence constraints and thereby complicate the combinatorial optimization problem.

To ensure that the generated compressions remain to be grammatical, additional syntactic constraints on the values of $\mathbf{d}$ need to be specified explicitly in the ILP model. We briefly describe them in the next independent subsection since they will also be used in our solution.

Syntactic Constraints

In previous work (Clarke and Lapata 2008), the following constraints have been introduced and shown to be useful for the task:

[1] This change is inevitable in ILP since simply using the product $\delta_i \delta_j \delta_k$ to denote a trigram (x_i, x_j, x_k) will make the objective function nonlinear. A specific binary variable t_{ijk} is required to keep linearity and additional linear constraints are needed to ensure consistency between t_{ijk} and $\mathbf{d}$.

Modifier Constraints Modifier constraints ensure that relationships between head words and their modifiers remain grammatical in the compression:

$$\delta_i - \delta_j \geq 0, \quad \forall i, j : x_j \in \text{cmod}(x_i), \tag{2}$$
$$\delta_i - \delta_j = 0, \quad \forall i, j : x_j \in \text{mmod}(x_i), \tag{3}$$

where cmod(x) denotes certain types of modifiers of x including non-clausal modifiers (`ncmod`) and determiners (`detmod`), while mmod(x) denotes negative modifiers (e.g. *not*, *never*) and possessive modifiers (e.g. *her*, *our*). Constraints (2) guarantee that if we include a certain-type modifier (e.g. non-clausal modifier, `ncmod`) in the compression (such as an adjective or a noun) then the head of the modifier must also be included. Constraints (3) ensure that the meaning of negations or possessions should be preserved in the compressions.

Argument-structure constraints There are a few intuitive constraints that take the overall sentence structure into account. The first constraint ensures that if a verb is present in the compression then so are its arguments (subjects and objects), and if any of the arguments are included in the compression then the verb must also be included:

$$\delta_i - \delta_j = 0, \quad \forall i, j : x_j \in \text{subj/obj}(x_i). \tag{4}$$

The second constraint forces the compression to contain at least one verb if the source sentence contains one as well:

$$\sum_{i: x_i \in verbs} \delta_i \geq 1. \tag{5}$$

The constraint entails that it is not possible to drop the main verb from any full sentence.

Other sentential constraints include those that are applied to prepositional phrases (`PP`) and subordinate clauses (`SUB`):

$$\delta_i - \delta_j = 0, \quad \forall i, j : x_j \in \text{PP/SUB}(x_i). \tag{6}$$

These constraints force the introducing term, mainly the preposition or subordinators such as *who* and *which*, to be included in the compression if any word from within the syntactic constituent is also included.

Meanwhile, there are constraints for coordinations (`cc`). If two head words are conjoined in the source sentence, then if they are included in the compression the coordinating conjunction must also be included:

$$(1 - \delta_i) + \delta_j \geq 1, \tag{7}$$
$$(1 - \delta_i) + \delta_k \geq 1, \tag{8}$$
$$\delta_i + (1 - \delta_j) + (1 - \delta_k) \geq 1, \tag{9}$$
$$\forall i, j, k : x_j \,\&\, x_k \text{ conjoined by } x_i. \tag{10}$$

Finally, anything within brackets should be treated comparatively less important in the source sentence:

$$\delta_i = 0, \forall i : x_i \in \text{ parenthesized.} \tag{11}$$

Discourse Constraints The discourse constraint introduced by Clarke and Lapata (2008) concerns personal pronouns as they should be included in the compression:

$$\delta_i = 1, \forall i : x_i \in \text{ personal pronouns.} \tag{12}$$

The Proposed Method

Regrouping Syntactic Constraints

The constraints introduced in the previous section are encoded as inequalities and equalities in an ILP formulation. We regroup the constraints according to the format rather than the encoded semantics, for the sake of our proposed algorithm. The format of different types of constraints can be regrouped into four specific categories:

- **Pre-specification constraints**: those requiring the variables to take pre-specified values, e.g. $\delta_i = 0$ for bracketed words and $\delta_i = 1$ for personal pronouns.
- **Head constraints**: for certain types of head-modifier relations, requiring $\delta_i - \delta_j \geq 0$.
- **Simultaneous constraints**: under certain types of conditions we require the head and the modifier to have the same decisions, i.e. $\delta_i - \delta_j = 0$. Also the coordination constraints will be classified as this type of constraints.
- **Non-local constraints**: mainly the verb constraint, i.e. $\sum_{i:x_i \in verbs} \delta_i \geq 1$.

In this study we address these types of constraints differently. These constraints can be absorbed into the inference procedure by our formulation.

Addressing the Constraints on Dependency Trees in Constrained Local Search

Our method is based on iterative greedy local search from a starting initialized solution. Given a randomized initialization of the bit vector $\mathbf{d}$, the inference process will traverse the dependency tree and try locally modifying the solution (flipping the value of δ_i along with possibly other variables to satisfy the constraints for δ_i) to see whether the objective function increases. During the traversal, syntactic constraints will be introduced at each relevant node.

We first deal with the **pre-specification constraints**. The solution is straightforward: update the bit vector to explicitly specify particular bits to be 0 or 1.

Since the other types of constraints are defined according to dependency relations or part-of-speech (POS) tags, our method can naturally take them into consideration when locally flipping the bits.

We perform a preorder traversal on the dependency tree, going top-down from the root to the leaves. At each node, we check for all local **simultaneous constraints** involving the current node x_i. A local search should be done while being consistent with these constraints, therefore multiple related simultaneous flips might be needed. Meanwhile, the **head constraints** can be injected almost trivially. Since we are going top-down, the decisions of children nodes will not affect the decision of the current node except in the case of simultaneous constraints which we have already dealt with. The only thing remains to be done is to check whether the current node has a head constraint with its direct head. If so, setting $\delta_i = 1$ for the current node will also trigger the bit for its head word to be 1 as well.

It is also easy to address the **non-local verb constraint**, we check the POS tag on the current node. If it is a verb, we will mark the current partial search as *legitimate*. The final solution will be the bit vector which scores the highest among those *legitimate* ones. Note that if no *legitimate* labels after a whole local search procedure, we know that there exist no verbs in the current sentence and the local search result can be directly returned.

We denote the above local flipping search procedure as LS_with_constraints [2]. This procedure will return a locally flipped version of $\mathbf{d}$. Local search will be implemented by comparing $\mathbf{d}^{(k)}$ and its flipped version. If we cannot improve the score for another whole traversal, we have reached a local optimum.

Randomized Local Search

Given such a local search algorithm, we can achieve local optimum from any initialized bit vector $\mathbf{d}^{(0)}$, since the objective function does not decrease in each iteration. All we need to achieve global optimum (or some good enough local optimum), is to set multiple random $\mathbf{d}$'s and run local search. Related randomized local search strategies have already been applied in various NLP tasks such as machine translation (Moore and Quirk 2008; Ravi and Knight 2010), language modeling (Deoras, Mikolov, and Church 2011), and dependency parsing (Zhang et al. 2014).

The whole randomized local search procedure is described in Algorithm 1. We name the procedure as randomized constrained greedy flipping.

Algorithm 1 Randomized constrained greedy flipping

```
Input: sentence $\mathbf{x}$ with dependency tree $\mathbf{t}$;
    scoring function: $F(\mathbf{x}, \mathbf{d})$
Output: Compression bit vector: $\tilde{\mathbf{d}}$
 1: Randomly initialize bit vector $\mathbf{d}^{(0)}$; $k \leftarrow 0$;
 2: $\mathbf{d}^{(0)} \leftarrow$ pre_specification($\mathbf{d}^{(0)}$)
 3: repeat
 4:     list $\leftarrow$ top-down node list of $\mathbf{d}^{(k)}$;
 5:     for each word/node in list do
 6:         $\mathbf{d}^{(k)} \leftarrow$ simul_constraints($\mathbf{x}, \mathbf{t}, \mathbf{d}^{(k)}$)
 7:         $\mathbf{d}' \leftarrow$ LS_with_constraints($\mathbf{x}, \mathbf{t}, \mathbf{d}^{(k)}$)
 8:         $\mathbf{d}^{(k+1)} \leftarrow \mathrm{argmax}_{\mathbf{d} \in \{\mathbf{d}^{(k)}, \mathbf{d}'\}} F(\mathbf{x}, \mathbf{d})$;
 9:         $k \leftarrow k + 1$;
10:     end for
11: until no change made in this iteration
12: return $\tilde{\mathbf{d}} = \mathbf{d}^{(k)}$
```

For a more clear description we give a simplified illustration. Figure 1 illustrates an example dependency tree for our explanation of the local search algorithm to compress the sentence $\mathbf{x}$ = *John gives him the correct answer.*, assuming a scoring function $F(\cdot)$. Suppose the initialization is $\mathbf{d}^{(0)} = 100111$. We first address the **pre-specification constraints** to set δ_3 to be 1 (the discourse constraint (12)). Then we traverse from the root word *gives* (x_2): considering the simultaneous constraints, we will be choosing from

[2]Due to the space limit we temporarily omit the pseudocode for summarizing this part.

$\mathbf{d}^{(0)} = 000111$ and $\mathbf{d}' = 111111$. Suppose the latter one scores higher, then $\mathbf{d}^{(1)} = 111111$. Meanwhile, since we are visiting a verb node, we mark the result as *legitimate*. After similar steps, when we visit the node *correct*, we will be choosing from $\mathbf{d}^{(5)} = 111111$ and $\mathbf{d}' = 111101$. Suppose $F(\mathbf{d}') > F(\mathbf{d}^{(5)})$, our local search result for this iteration will be $\mathbf{d}^{(6)} = 111101$, corresponding to preserving all words but the less informative adjective *correct*.

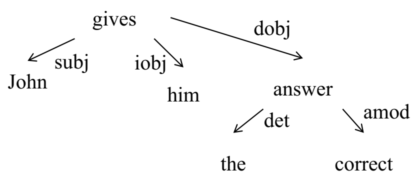

Figure 1: An example sentence to be compressed

It is easy to see that the local search process can be easily done in parallel. We leave the exploration on how much a crafted parallel implementation can boost the acceleration as future work. Also, caching the scores for queried bit vectors can make the process more efficient.

Score Definition

Our proposed solution can be used for any scoring function without strong assumptions, as long as it can be evaluated. In this work we study several scoring schemes with some of them integrating multiple local structures.

Importance Weights Assignment For word importance weight $w(x_i)$ in (1), there are many ways that can be defined. In this study we use the formula introduced by (Clarke and Lapata 2008) which is a modification of the significance score of (Hori and Furui 2003):

$$w(x_i) = f_i \log \frac{F_A}{F_i}, \tag{13}$$

where f_i and F_i are the frequency of x_i in the document and corpus respectively, F_a is the sum of all topic words in the corpus, l is the number of clause constituents above x_i, and N is the deepest level of clause embedding. F_a and F_i are estimated from a large document collection, f_i is document-specific, whereas $\frac{l}{N}$ is sentence specific.

Scoring for Rich Local Structures Any scores or weights defined locally on words, ngrams or dependency edges can be used for our method, and in principle for the ILP models with the introduction of many additional variables and constraints. In this study we explore the usage of trigram scores and local discriminative scores on compression bigrams, to form consistency with previous work (Clarke and Lapata 2008).

Trigram scores Following previous work, we utilize trigram scores in the form of log probabilities from a pre-trained language model to enhance local smoothness.

Structured Discriminative Scores In ILP-based models proposed by Clarke and Lapata (2008), alternative scores derived from the structured discriminative model presented by McDonald (2006) have also been explored. This model uses a large-margin learning framework (MIRA) coupled with a feature set defined on compression bigrams and syntactic structure. Given learned weights, the model generate compressions with:

$$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \quad \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}), \tag{14}$$

where $\mathcal{Y}(\mathbf{x})$ denotes the candidate set of compressions for the original sentence $\mathbf{x}$. This decoding procedure can be efficiently performed with Viterbi-like dynamic programming if $\mathbf{f}$ is defined locally. Following previous work (McDonald 2006; Clarke and Lapata 2008), in this work we use bigrams in the target compression as the local structure.

Any training strategy that is applicable for typical global linear models can be adopted. We trained the model using the structured perceptron (Collins 2002) modified with AdaGrad updates (Duchi, Hazan, and Singer 2011) [3]:

$$\mathbf{w}_i^{t+1} \leftarrow \mathbf{w}_i^t - \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} (\mathbf{s}_i^\tau)^2}} \mathbf{s}_i^t, \tag{15}$$

where $\mathbf{s}^t = \mathbf{f}(\mathbf{x}^t, \mathbf{y}_*^t) - \mathbf{f}(\mathbf{x}^t, \mathbf{y}^t)$ is the subgradient for instance t and η is the learning rate that is set to be constant 1.0 in this work. We take the discriminative weights after five training epochs as they perform well on the development sets.

Length Control

In the original ILP formulations by Clarke and Lapata (2008), the control over the lengths of the generated compressions are directly encoded in yet another linear constraint $\sum_i \delta_i \geq b$. Since it is unfair to set hard length controlling constant for every sentence, we use an alternative way by introducing a parameter $\alpha > 0$ to control the effect of word weights in the objective function (1), slightly changing it into:

$$F_{\mathbf{x},\alpha}(\mathbf{d}) = S(\mathbf{x}, \mathbf{d}) + \alpha \sum_{i=1}^{n} \delta_i \cdot w(x_i). \tag{16}$$

Since the weights are positive (while the log probability language model scores in $S(\mathbf{x}, \mathbf{d})$ are negative), a larger α will encourage the selection of more words. Using this length control strategy, there will be no change to the proposed algorithm other than evaluating $F(\cdot)$.

Experiments

Data

We evaluate our methods on two corpora that were annotated by human annotators [4]. One was created on sentences sampled from the British National Corpus (BNC) and the

[3]We also tried MIRA used by McDonald (2006) but observed slightly inferior performance, compared with AdaGrad perceptron.

[4]Available at http://jamesclarke.net/research/resources

American News Text Corpus, while the other was human-annotated compressions on broadcast news. Therefore we will refer to these two corpora as Written and Spoken respectively. We split the datasets into training, development and test sets according to (Galanis and Androutsopoulos 2010).

To get dependency parses and implement checkers for syntactic constraints, all sentences are parsed with the Stanford parser [5].

Evaluation Metrics

We evaluate the compression results by both automatic metrics and manual ratings.

For automatic evaluation, there are two metrics to be considered in the sentence compression task. The compression ratio calculates the ratio of compression lengths and original lengths, reflecting the difference before and after being compressed. The other measure is the F1-score of grammatical relations of generated compressions against the gold-standard compression. The dependency relations are provided by a dependency parser with grammatical labels. This measure has been generally accepted in previous works of sentence compression. Clarke and Lapata (2006) showed that it correlates well with human ratings. To avoid overfitting the parsing errors from the Stanford Parser during validation, we chose MaltParser [6], a well-known transition-based dependency parser, at the evaluation stage.

For fair comparisons, we tune relevant parameters on the development set to make all system producing outputs with similar compression ratio which is close to the gold-standard manual compressions.

We also conduct manual evaluation. As manually evaluating the whole test set requires enormous efforts, following previous work, we sample a subset of 30 sentences from each dataset for manual rating. Three annotators who are fluent in English were asked to rate each candidate for the same long sentence. Ratings should be in the form of 1-5 scores for each compression. Compressions with better grammaticality and importance preservation should be assigned with higher scores.

Methods in Comparison

We report results from the following systems:

- **McDonald06**: We reimplemented the structured discriminative model by McDonald (2006) [7].
- **GA10**: The system proposed by Galanis and Androutsopoulos (2010), essentially a two-stage method to rerank compressions generated by a discriminative maximum entropy model.
- **T3**: Tree transduction models proposed by Cohn and Lapata (2009) which learns parse tree transduction rules from a parallel corpus using a large margin method.
- **ILP(LM)**: Using ILP solvers to solve 1 with syntactic constraints, with the scores $S(\cdot)$ defined as log probabilities from a trigram language model.
- **ILP(Disc)**: Similar to the above with the scores $S(\cdot)$ defined as structured discriminative models. This model, along with **ILP(LM)**, can be treated as reimplementations of ILP models proposed by Clarke and Lapata (2008).
- **ILP(LM+Disc)**: Similar ILP formulations with the scores combined from the above two systems. Note that there will be indicator variables for multiple types of local structures, which simultaneously increase the number of variables and consistency constraints, and hence the overall complexity.
- **RCGF**: The counterparts of ILP models using our proposed randomized constrained greedy flipping algorithm with $K = \min\{300, 2^{|\mathbf{x}|}\}$ different initializations.

Most of the systems in comparison include scores derived from a language model. We train a trigram language model on the Reuters Corpus (Volume 1) [8] with modified Kneser-Ney smoothing, using the widely used tool SRILM [9]. All ILPs are solved using a state-of-the-art solver GLPK [10].

Results

Table 1 summarizes the results from our compression experiments on the two corpora. The rightmost column lists averaged inference time per sentence. The difference of GR-F1 between the top group and the rest, along with the difference of time between ILP and the rest, have passed multiple testing for the significance level $p < 0.01$. Note that to show the superiority in average, we have already excluded a few extreme cases that caused the ILP solver too much time (running more than a minute for one single sentence) merely when calculating the average cost for ILP models.

From Table 1 we can obviously observe that ILP models with explicit syntactic constraints outperforms other baselines. For different objective functions in ILP models, only using discriminative scores performs the worst, while adding them on language model scores can boost the performance.

In terms of running time, the McDonald06 system performs an order of magnitude faster since the inference procedure only involves an efficient dynamic programming (Viterbi) process. Compare to ILP counterparts, our proposed greedy flipping algorithm makes significantly much faster inference while giving almost identical GR-F1 scores.

Table 2 displays results for manual evaluation on the sampled 30 sentences in each dataset. Models with explicit constraints (ILP and RCGF) perform significantly better in terms of both grammaticality and importance preservation, compared with the other systems. There hardly exists any difference between the quality of the generated compressions from ILP models and those from our method. We find that the compressions for the sampled sentences are identical for Written corpus [11].

[5]http://nlp.stanford.edu/software/lex-parser.html

[6]http://www.maltparser.org/

[7]For consistent comparisons with other systems, our reimplementation does not include the k-best inference strategy presented by McDonald (2006) for learning with MIRA.

[8]http://trec.nist.gov/data/reuters/reuters.html

[9]http://www-speech.sri.com/projects/srilm/

[10]http://www.gnu.org/software/glpk/

[11]Since the compressions are displayed to the annotator simulta-

Written	CR(%)	GR-F1(%)	Time (s)
McDonald06	70.3	52.4	0.02
GA10	71.5	60.2	0.77
T3	70.4	58.8	0.75
ILP(LM)	71.2	64.5	1.23
ILP(Disc)	70.8	62.0	1.22
ILP(LM+Disc)	71.8	66.5	1.72
RCGF(LM)	71.3	64.5	0.25
RCGF(Disc)	70.6	61.8	0.27
RCGF(LM+Disc)	71.7	66.4	0.28
Gold-Standard	71.4	100.0	-
Spoken	**CR(%)**	**GR-F1(%)**	**Time (s)**
McDonald06	69.5	50.6	0.02
GA10	71.7	59.2	0.75
T3	75.5	59.5	0.76
ILP(LM)	71.3	61.5	1.05
ILP(Disc)	70.2	56.2	1.06
ILP(LM+Disc)	72.6	66.2	1.49
RCGF(LM)	71.3	61.5	0.22
RCGF(Disc)	70.2	55.9	0.22
RCGF(LM+Disc)	72.7	66.2	0.25
Gold-Standard	72.4	100.0	-

Table 1: Results of automatic evaluation

Written	**GR.**	**Imp.**	**CR(%)**
McDonald06	$3.68^{\dagger *}$	$3.55^{\dagger *}$	70.7
GA10	$3.96^{\dagger *}$	$3.57^{\dagger *}$	71.6
T3	$4.22^{\dagger *}$	$3.69^{\dagger *}$	72.0
ILP(LM+Disc)	4.56	4.14	72.2
RCGF(LM+Disc)	4.56	4.14	72.2
Gold-Standard	4.86	4.80	72.7
Spoken	**GR.**	**Imp.**	**CR(%)**
McDonald06	$3.77^{\dagger *}$	$3.49^{\dagger *}$	72.0
GA10	$4.03^{\dagger *}$	$3.81^{\dagger *}$	72.3
T3	$4.17^{\dagger *}$	$3.68^{\dagger *}$	74.0
ILP(LM+Disc)	4.38	4.03	72.4
RCGF(LM+Disc)	4.38	4.02	72.3
Gold-Standard	4.82	4.86	73.4

Table 2: Results of manual ratings. (†: sig. diff. from ILP; *: sig. diff. from RCGF, for $p < 0.01$)

Dataset	Length ≤ 15	Length > 15
Written	100	98.1
Spoken	100	96.4

Table 3: Fractions (%) of sentences that our method find the optimal solution

Analysis and Discussion

Our proposed greedy flipping algorithm do not have large number of variables as in ILP models, while most decisions are made locally during a linear-time preorder tree traversal. Due to the effectiveness in each iteration, our method still remains to be fast although requiring multiple random restarts. Therefore, for optimizing the same objective function with the same constraints, our proposed algorithm behaves much faster than ILP models while providing almost the same competitive results. As the random search procedure can be proceeded independently, more parallelized implementation will further improve our inference algorithm.

Due to the complex nature of our problem structure with different types of constraints, it is difficult to give a formal mathematical analysis for worst-case convergence bound.

To shed some light on the reason why simple random local search process can indeed work, we perform some statistics first. Table 3 contains some statistics over the success rate, i.e. the proportion of sentences whose global optimal solution (obtained via ILP) can be found efficiently using our method, showing that for short sentences (no more than 15 words) it is rather easy to achieve global optimal. For longer sentences we can also expect our algorithm to behave nicely, with very few long and complex sentences reaching sub-optimal solutions.

We also draw a figure (Figure 2, the solid blue path) to depict the magnitude of objective function (ratio of currently found best solution against the global optimal, y-axis), proceeding with iterations (x-axis). The example is sampled from the shorter sentences. We can see that the path jumped for very few times and reach the global optimum quickly. We also include one particular long sentence which our method fails to find the exact global optimum (Figure 2, dotted path), jumping more often than the simpler sentence.

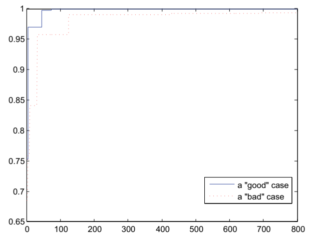

Figure 2: Examples of running paths

Therefore we conjecture that the success of our simple randomized constrained local search algorithm is mainly due to the fact that the number of (good) local optimal points are rather small, perhaps at the order of tens, compared with the overall search space in size of $2^{|\mathbf{x}|}$ for bit vectors. This property is particularly significant for shorter sentences with simpler structures. Some of the local solutions might be close to the global optimum as well.

Conclusion and Future Work

In this study we propose a randomized constrained local search algorithm for sentence compression, which can naturally integrate syntactic constraints and generate equally competitive compressions as ILP-based models, while being way much faster.

neously, though being randomly shuffled and made "anonymous" for each sentence, same compressions will receive the same ratings in normal cases.

For future study we would like to explore the potential effectiveness of discriminative scores given training data in larger scale, as some efforts have focused on extracting parallel corpus for sentence compression (Filippova and Altun 2013). Meanwhile, It is also worthwhile to study what kinds of initializations may lead to better optimal or sub-optimal solutions.

Our local search process resembles Metropolis-Hastings or Gibbs sampling for Markov Chain Monte Carlo inference in Bayesian probabilistic models. We would like to study sampling-based methods and make comparisons as well.

Acknowledgments

We thank all reviewers for helpful comments on an earlier draft of this paper. This work was supported by National Natural Science Foundation of China (61331011), National Hi-Tech Research and Development Program (863 Program) of China (2015AA015403) and IBM Global Faculty Award Program. Xiaojun Wan is the corresponding author.

References

Clarke, J., and Lapata, M. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the COLING/ACL*, 377–384. Association for Computational Linguistics.

Clarke, J., and Lapata, M. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research* 31:273–381.

Cohn, T., and Lapata, M. 2007. Large margin synchronous generation and its application to sentence compression. In *Proceedings of EMNLP-CoNLL*, 73–82. Prague, Czech Republic: Association for Computational Linguistics.

Cohn, T., and Lapata, M. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research* 34:637–674.

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

Deoras, A.; Mikolov, T.; and Church, K. 2011. A fast re-scoring strategy to capture long-distance dependencies. In *Proceedings of EMNLP*, 1116–1127. Edinburgh, Scotland, UK.: Association for Computational Linguistics.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research* 12:2121–2159.

Filippova, K., and Altun, Y. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of EMNLP*, 1481–1491. Seattle, Washington, USA: Association for Computational Linguistics.

Filippova, K., and Strube, M. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, 25–32. Association for Computational Linguistics.

Filippova, K.; Alfonseca, E.; Colmenares, C. A.; Kaiser, L.; and Vinyals, O. 2015. Sentence compression by deletion with LSTMs. In *Proceedings of EMNLP*, 360–368. Lisbon, Portugal: Association for Computational Linguistics.

Galanis, D., and Androutsopoulos, I. 2010. An extractive supervised two-stage method for sentence compression. In *HLT-NAACL*, 885–893. Los Angeles, California: Association for Computational Linguistics.

Hori, C., and Furui, S. 2003. A new approach to automatic speech summarization. *Multimedia, IEEE Transactions on* 5(3):368–378.

Huang, M.; Shi, X.; Jin, F.; and Zhu, X. 2012. Using first-order logic to compress sentences. In *AAAI*.

Knight, K., and Marcu, D. 2000. Statistics-based summarization - step one: Sentence compression. In *AAAI/IAAI*, 703–710.

McDonald, R. T. 2006. Discriminative sentence compression with soft syntactic evidence. In *EACL*.

Moore, R. C., and Quirk, C. 2008. Random restarts in minimum error rate training for statistical machine translation. In *Proceedings of COLING 2008*, 585–592. Manchester, UK: Coling 2008 Organizing Committee.

Narayan, S., and Gardent, C. 2014. Hybrid simplification using deep semantics and machine translation. In *Proceedings of the ACL*, 435–445. Baltimore, Maryland: Association for Computational Linguistics.

Nomoto, T. 2009. A comparison of model free versus model intensive approaches to sentence compression. In *Proceedings of EMNLP*, 391–399. Singapore: Association for Computational Linguistics.

Ravi, S., and Knight, K. 2010. Does giza++ make search errors? *Computational Linguistics* 36(3):295–302.

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*, 379–389. Lisbon, Portugal: Association for Computational Linguistics.

Thadani, K., and McKeown, K. 2013. Sentence compression with joint structural inference. In *Proceedings of CoNLL*, 65–74. Sofia, Bulgaria: Association for Computational Linguistics.

Thadani, K. 2014. Approximation strategies for multi-structure sentence compression. In *Proceedings of the ACL*, 1241–1251. Baltimore, Maryland: Association for Computational Linguistics.

Turner, J., and Charniak, E. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of the ACL*, 290–297. Association for Computational Linguistics.

Woodsend, K., and Lapata, M. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of EMNLP*, 409–420. Association for Computational Linguistics.

Xu, W.; Napoles, C.; Pavlick, E.; Chen, Q.; and Callison-Burch, C. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*.

Yamangil, E., and Shieber, S. M. 2010. Bayesian synchronous tree-substitution grammar induction and its application to sentence compression. In *Proceedings of the ACL*, 937–947. Association for Computational Linguistics.

Yao, J.-g.; Wan, X.; and Xiao, J. 2014. Joint decoding of tree transduction models for sentence compression. In *Proceedings of EMNLP*, 1828–1833. Doha, Qatar: Association for Computational Linguistics.

Yoshikawa, K.; Hirao, T.; Iida, R.; and Okumura, M. 2012. Sentence compression with semantic role constraints. In *Proceedings of the ACL*, 349–353. Association for Computational Linguistics.

Zhang, Y.; Lei, T.; Barzilay, R.; and Jaakkola, T. 2014. Greed is good if randomized: New inference for dependency parsing. In *Proceedings of EMNLP*, 1013–1024. Doha, Qatar: Association for Computational Linguistics.