Approximation Strategies for Multi-Structure Sentence Compression

Kapil Thadani
Department of Computer Science
Columbia University
New York, NY 10025, USA
kapil@cs.columbia.edu

Abstract

Sentence compression has been shown to benefit from joint inference involving both n-gram and dependency-factored objectives but this typically requires expensive integer programming. We explore instead the use of Lagrangian relaxation to decouple the two subproblems and solve them separately. While dynamic programming is viable for bigram-based sentence compression, finding optimal compressed trees within graphs is NP-hard. We recover approximate solutions to this problem using LP relaxation and maximum spanning tree algorithms, yielding techniques that can be combined with the efficient bigram-based inference approach using Lagrange multipliers. Experiments show that these approximation strategies produce results comparable to a state-of-the-art integer linear programming formulation for the same joint inference task along with a significant improvement in runtime.

1 Introduction

Sentence compression is a text-to-text generation task in which an input sentence must be transformed into a shorter output sentence which accurately reflects the meaning in the input and also remains grammatically well-formed. The compression task has received increasing attention in recent years, in part due to the availability of datasets such as the Ziff-Davis corpus (Knight and Marcu, 2000) and the Edinburgh compression corpora (Clarke and Lapata, 2006), from which the following example is drawn.

Original: In 1967 Chapman, who had cultivated a conventional image with his ubiquitous tweed jacket and pipe, by his own later admission stunned a party attended by his friends and future Python colleagues by coming out as a homosexual.

Compressed: In 1967 Chapman, who had cultivated a conventional image, stunned a party by coming out as a homosexual.

Following an assumption often used in compression systems, the compressed output in this corpus is constructed by dropping tokens from the input sentence without any paraphrasing or reordering.[1]

A number of diverse approaches have been proposed for deletion-based sentence compression, including techniques that assemble the output text under an n-gram factorization over the input text (McDonald, 2006; Clarke and Lapata, 2008) or an arc factorization over input dependency parses (Filippova and Strube, 2008; Galanis and Androutsopoulos, 2010; Filippova and Altun, 2013). Joint methods have also been proposed that invoke integer linear programming (ILP) formulations to simultaneously consider multiple structural inference problems—both over n-grams and input dependencies (Martins and Smith, 2009) or n-grams and all possible dependencies (Thadani and McKeown, 2013). However, it is well-established that the utility of ILP for optimal inference in structured problems is often outweighed by the worst-case performance of ILP solvers on large problems without unique integral solutions. Furthermore, approximate solutions can often be adequate for real-world generation systems, particularly in the presence of linguistically-motivated constraints such as those described by Clarke and Lapata (2008), or domain-specific

[1] This is referred to as *extractive compression* by Cohn and Lapata (2008) & Galanis and Androutsopoulos (2010) following the terminology used in document summarization.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1241–1251,
Baltimore, Maryland, USA, June 23-25 2014.

pruning strategies such as the use of sentence templates to constrain the output.

In this work, we develop approximate inference strategies to the joint approach of Thadani and McKeown (2013) which trade the optimality guarantees of exact ILP for faster inference by separately solving the n-gram and dependency subproblems and using Lagrange multipliers to enforce consistency between their solutions. However, while the former problem can be solved efficiently using the dynamic programming approach of McDonald (2006), there are no efficient algorithms to recover maximum weighted non-projective subtrees in a general directed graph. Maximum spanning tree algorithms, commonly used in non-projective dependency parsing (McDonald et al., 2005), are not easily adaptable to this task since the maximum-weight subtree is not necessarily a part of the maximum spanning tree.

We therefore consider methods to recover approximate solutions for the subproblem of finding the maximum weighted subtree in a graph, common among which is the use of a linear programming relaxation. This linear program (LP) appears empirically tight for compression problems and our experiments indicate that simply using the non-integral solutions of this LP in Lagrangian relaxation can empirically lead to reasonable compressions. In addition, we can recover approximate solutions to this problem by using the Chu-Liu Edmonds algorithm for recovering maximum spanning trees (Chu and Liu, 1965; Edmonds, 1967) over the relatively sparse subgraph defined by a solution to the relaxed LP. Our proposed approximation strategies are evaluated using automated metrics in order to address the question: under what conditions should a real-world sentence compression system implementation consider exact inference with an ILP or approximate inference? The contributions of this work include:

- An empirically-useful technique for approximating the maximum-weight subtree in a weighted graph using LP-relaxed inference.
- Multiple approaches to generate good approximate solutions for joint multi-structure compression, based on Lagrangian relaxation to enforce equality between the sequential and syntactic inference subproblems.
- An analysis of the tradeoffs incurred by joint approaches with regard to runtime as well as performance under automated measures.

2 Multi-Structure Sentence Compression

Even though compression is typically formulated as a token deletion task, it is evident that dropping tokens independently from an input sentence will likely not result in fluent and meaningful compressive text. Tokens in well-formed sentences participate in a number of syntactic and semantic relationships with other tokens, so one might expect that accounting for heterogenous structural relationships between tokens will improve the coherence of the output sentence. Furthermore, much recent work has focused on the challenge of joint sentence extraction and compression, also known as *compressive summarization* (Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Almeida and Martins, 2013; Li et al., 2013; Qian and Liu, 2013), in which questions of efficiency are paramount due to the larger problems involved; however, these approaches largely restrict compression to pruning parse trees, thereby imposing a dependency on parser performance. We focus in this work on a sentence-level compression system to approximate the ILP-based inference of Thadani and McKeown (2013) which does not restrict compressions to follow input parses but permits the generation of novel dependency relations in output compressions.

The rest of this section is organized as follows: §2.1 provies an overview of the joint sequential and syntactic objective for compression from Thadani and McKeown (2013) while §2.2 discusses the use of Lagrange multipliers to enforce consistency between the different structures considered. Following this, §2.3 discusses a dynamic program to find maximum weight bigram subsequences from the input sentence, while §2.4 covers LP relaxation-based approaches for approximating solutions to the problem of finding a maximum-weight subtree in a graph of potential output dependencies. Finally, §2.5 discusses the features and model training approach used in our experimental results which are presented in §3.

2.1 Joint objective

We begin with some notation. For an input sentence S comprised of n tokens including duplicates, we denote the set of tokens in S by $T \triangleq \{t_i : 1 \leq i \leq n\}$. Let C represent a compression of S and let $x_i \in \{0, 1\}$ denote an indicator variable whose value corresponds to whether token $t_i \in T$ is present in the compressed sentence

C. In addition, we define bigram indicator variables $y_{ij} \in \{0, 1\}$ to represent whether a particular order-preserving bigram[2] $\langle t_i, t_j \rangle$ from S is present as a contiguous bigram in C as well as dependency indicator variables $z_{ij} \in \{0, 1\}$ corresponding to whether the dependency arc $t_i \rightarrow t_j$ is present in the dependency parse of C. The score for a given compression C can now be defined to factor over its tokens, n-grams and dependencies as follows.

$$\begin{aligned} score(C) \;=\; & \sum_{t_i \in T} x_i \cdot \theta_{\text{tok}}(t_i) \\ & + \sum_{\substack{t_i \in T \cup \{\text{START}\}, \\ t_j \in T \cup \{\text{END}\}}} y_{ij} \cdot \theta_{\text{bgr}}(\langle t_i, t_j \rangle) \\ & + \sum_{\substack{t_i \in T \cup \{\text{ROOT}\}, \\ t_j \in T}} z_{ij} \cdot \theta_{\text{dep}}(t_i \rightarrow t_j) \end{aligned} \quad (1)$$

where θ_{tok}, θ_{bgr} and θ_{dep} are feature-based scoring functions for tokens, bigrams and dependencies respectively. Specifically, each $\theta_v(\cdot) \equiv \mathbf{w}_v^\top \boldsymbol{\phi}_v(\cdot)$ where $\boldsymbol{\phi}_v(\cdot)$ is a feature map for a given variable type $v \in \{\text{tok, bgr, dep}\}$ and $\mathbf{w}_v$ is the corresponding vector of learned parameters.

The inference task involves recovering the highest scoring compression C^* under a particular set of model parameters $\mathbf{w}$.

$$\begin{aligned} C^* &= \arg\max_C \; score(C) \\ &= \arg\max_{\mathbf{x},\mathbf{y},\mathbf{z}} \; \mathbf{x}^\top \boldsymbol{\theta}_{\text{tok}} + \mathbf{y}^\top \boldsymbol{\theta}_{\text{bgr}} + \mathbf{z}^\top \boldsymbol{\theta}_{\text{dep}} \end{aligned} \quad (2)$$

where the incidence vector $\mathbf{x} \triangleq \langle x_i \rangle_{t_i \in T}$ represents an entire token configuration over T, with $\mathbf{y}$ and $\mathbf{z}$ defined analogously to represent configurations of bigrams and dependencies. $\boldsymbol{\theta}_v \triangleq \langle \theta_v(\cdot) \rangle$ denotes a corresponding vector of scores for each variable type v under the current model parameters. In order to recover meaningful compressions by optimizing (2), the inference step must ensure:

1. The configurations $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are *consistent* with each other, i.e., all configurations cover the same tokens.
2. The structural configurations $\mathbf{y}$ and $\mathbf{z}$ are *non-degenerate*, i.e, the bigram configuration $\mathbf{y}$ represents an acyclic path while the dependency configuration $\mathbf{z}$ forms a tree.

These requirements naturally rule out simple approximate inference formulations such as search-based approaches for the joint objective.[3] An ILP-based inference solution is demonstrated in Thadani and McKeown (2013) that makes use of linear constraints over the boolean variables x_i, y_{ij} and z_{ij} to guarantee consistency, as well as auxiliary real-valued variables and constraints representing the flow of commodities (Magnanti and Wolsey, 1994) in order to establish structure in $\mathbf{y}$ and $\mathbf{z}$. In the following section, we propose an alternative formulation that exploits the modularity of this joint objective.

2.2 Lagrangian relaxation

Dual decomposition (Komodakis et al., 2007) and Lagrangian relaxation in general are often used for solving joint inference problems which are decomposable into individual subproblems linked by equality constraints (Koo et al., 2010; Rush et al., 2010; Rush and Collins, 2011; DeNero and Macherey, 2011; Martins et al., 2011; Das et al., 2012; Almeida and Martins, 2013). This approach permits sub-problems to be solved separately using problem-specific efficient algorithms, while consistency over the structures produced is enforced through Lagrange multipliers via iterative optimization. Exact solutions are guaranteed when the algorithm converges on a consistent primal solution, although this convergence itself is not guaranteed and depends on the tightness of the underlying LP relaxation. The primary advantage of this technique is the ability to leverage the underlying structure of the problems in inference rather than relying on a generic ILP formulation while still often producing exact solutions.

The multi-structure inference problem described in the previous section seems in many ways to be a natural fit to such an approach since output scores factor over different types of structure that comprise the output compression. Even if ILP-based approaches perform reasonably at the scale of single-sentence compression problems, the exponential worst-case complexity of general-purpose ILPs will inevitably pose challenges when scaling up to (a) handle larger inputs, (b) use higher-order structural fragments, or (c) incorporate additional models.

[2] Although Thadani and McKeown (2013) is not restricted to bigrams or order-preserving n-grams, we limit our discussion to this scenario as it also fits the assumptions of McDonald (2006) and the datasets of Clarke and Lapata (2006).

[3] This work follows Thadani and McKeown (2013) in recovering non-projective trees for inference. However, recovering projective trees is tractable when a total ordering of output tokens is assumed. This will be addressed in future work.

Consider once more the optimization problem characterized by (2) The two structural problems that need to be solved in this formulation are the extraction of a maximum-weight acyclic subsequence of bigrams $\mathbf{y}$ from the lattice of all order-preserving bigrams from S and the recovery of a maximum-weight directed subtree $\mathbf{z}$. Let $\boldsymbol{\alpha}(\mathbf{y}) \in \{0,1\}^n$ denote the incidence vector of tokens contained in the n-gram sequence $\mathbf{y}$ and $\boldsymbol{\beta}(\mathbf{z}) \in \{0,1\}^n$ denote the incidence vector of words contained in the dependency tree $\mathbf{z}$. We can now rewrite the objective in (2) while enforcing the constraint that the words contained in the sequence $\mathbf{y}$ are the same as the words contained in the tree $\mathbf{z}$, i.e., $\boldsymbol{\alpha}(\mathbf{y}) = \boldsymbol{\beta}(\mathbf{z})$, by introducing a vector of Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^n$. In addition, the token configuration $\mathbf{x}$ can be rewritten in the form of a weighted combination of $\boldsymbol{\alpha}(\mathbf{y})$ and $\boldsymbol{\beta}(\mathbf{z})$ to ensure its consistency with $\mathbf{y}$ and $\mathbf{z}$. This results in the following Lagrangian:

$$\begin{aligned} L(\boldsymbol{\lambda}, \mathbf{y}, \mathbf{z}) = \mathbf{y}^\top \boldsymbol{\theta}_{\text{bgr}} + \mathbf{z}^\top \boldsymbol{\theta}_{\text{dep}} \\ + \boldsymbol{\theta}_{\text{tok}}^\top (\psi \cdot \boldsymbol{\alpha}(\mathbf{y}) + (1-\psi) \cdot \boldsymbol{\beta}(\mathbf{z})) \\ + \boldsymbol{\lambda}^\top (\boldsymbol{\alpha}(\mathbf{y}) - \boldsymbol{\beta}(\mathbf{z})) \end{aligned} \tag{3}$$

Finding the $\mathbf{y}$ and $\mathbf{z}$ that maximize this Lagrangian above yields a dual objective, and the dual problem corresponding to the primal objective specified in (2) is therefore the minimization of this objective over the Lagrange multipliers $\boldsymbol{\lambda}$.

$$\begin{aligned} &\min_{\boldsymbol{\lambda}} \max_{\mathbf{y},\mathbf{z}} L(\boldsymbol{\lambda}, \mathbf{y}, \mathbf{z}) \\ =&\min_{\boldsymbol{\lambda}} \max_{\mathbf{y}} \mathbf{y}^\top \boldsymbol{\theta}_{\text{bgr}} + (\boldsymbol{\lambda} + \psi \cdot \boldsymbol{\theta}_{\text{tok}})^\top \boldsymbol{\alpha}(\mathbf{y}) \\ &+ \max_{\mathbf{z}} \mathbf{z}^\top \boldsymbol{\theta}_{\text{dep}} - (\boldsymbol{\lambda} + (\psi - 1) \cdot \boldsymbol{\theta}_{\text{tok}})^\top \boldsymbol{\beta}(\mathbf{z}) \\ =&\min_{\boldsymbol{\lambda}} \max_{\mathbf{y}} f(\mathbf{y}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta}) \\ &+ \max_{\mathbf{z}} g(\mathbf{z}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta}) \end{aligned} \tag{4}$$

This can now be solved with the iterative subgradient algorithm illustrated in Algorithm 1. In each iteration i, the algorithm solves for $\mathbf{y}^{(i)}$ and $\mathbf{z}^{(i)}$ under $\boldsymbol{\lambda}^{(i)}$, then generates $\boldsymbol{\lambda}^{(i+1)}$ to penalize inconsistencies between $\boldsymbol{\alpha}(\mathbf{y}^{(i)})$ and $\boldsymbol{\beta}(\mathbf{z}^{(i)})$. When $\boldsymbol{\alpha}(\mathbf{y}^{(i)}) = \boldsymbol{\beta}(\mathbf{z}^{(i)})$, the resulting primal solution is exact, i.e., $\mathbf{y}^{(i)}$ and $\mathbf{z}^{(i)}$ represent the optimal structures under (2). Otherwise, if the algorithm starts oscillating between a few primal solutions, the underlying LP must have a non-integral solution in which case approximation heuristics can be employed.[4] The application of this Lagrangian relaxation strategy is contingent upon the existence of algorithms to solve the maximization subproblems for $f(\mathbf{y}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta})$ and $g(\mathbf{z}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta})$. The following sections discuss our approach to these problems.

Algorithm 1 Subgradient-based joint inference

Input: scores $\boldsymbol{\theta}$, ratio ψ, repetition limit l_{max}, iteration limit i_{max}, learning rate schedule $\boldsymbol{\eta}$
Output: token configuration $\mathbf{x}$

$\boldsymbol{\lambda}^{(0)} \leftarrow \langle 0 \rangle^n$
$M \leftarrow \varnothing, M_{\text{repeats}} \leftarrow \varnothing$
for iteration $i < i_{\text{max}}$ **do**
  $\hat{\mathbf{y}} \leftarrow \arg\max_{\mathbf{y}} f(\mathbf{y}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta})$
  $\hat{\mathbf{z}} \leftarrow \arg\max_{\mathbf{z}} g(\mathbf{z}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta})$
  if $\boldsymbol{\alpha}(\hat{\mathbf{y}}) = \boldsymbol{\beta}(\hat{\mathbf{z}})$ **then return** $\boldsymbol{\alpha}(\hat{\mathbf{y}})$
  if $\boldsymbol{\alpha}(\hat{\mathbf{y}}) \in M$ **then**
    $M_{\text{repeats}} \leftarrow M_{\text{repeats}} \cup \{\boldsymbol{\alpha}(\hat{\mathbf{y}})\}$
  if $\boldsymbol{\beta}(\hat{\mathbf{z}}) \in M$ **then**
    $M_{\text{repeats}} \leftarrow M_{\text{repeats}} \cup \{\boldsymbol{\beta}(\hat{\mathbf{z}})\}$
  if $|M_{\text{repeats}}| \geq l_{\text{max}}$ **then break**
  $M \leftarrow M \cup \{\boldsymbol{\alpha}(\hat{\mathbf{y}}), \boldsymbol{\beta}(\hat{\mathbf{z}})\}$
  $\boldsymbol{\lambda}^{(i+1)} \leftarrow \boldsymbol{\lambda}^{(i)} - \eta_i \left(\boldsymbol{\alpha}(\hat{\mathbf{y}}) - \boldsymbol{\beta}(\hat{\mathbf{z}})\right)$
return $\arg\max_{\mathbf{x} \in \mathbf{M}_{\text{repeats}}} score(\mathbf{x})$

2.3 Bigram subsequences

McDonald (2006) provides a Viterbi-like dynamic programming algorithm to recover the highest-scoring sequence of order-preserving bigrams from a lattice, either in unconstrained form or with a specific length constraint. The latter requires a dynamic programming table $Q[i][r]$ which represents the best score for a compression of length r ending at token i. The table can be populated using the following recurrence:

$$\begin{aligned} Q[i][1] &= score(S, \text{START}, i) \\ Q[i][r] &= \max_{j<i} Q[j][r-1] + score(S, i, j) \\ Q[i][R+1] &= Q[i][R] + score(S, i, \text{END}) \end{aligned}$$

where R is the required number of output tokens and the scoring function is defined as

$$score(S, i, j) \triangleq \theta_{\text{bgr}}(\langle t_i, t_j \rangle) + \lambda_j + \psi \cdot \theta_{\text{tok}}(t_j)$$

so as to solve $f(\mathbf{y}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta})$ from (4). This approach requires $O(n^2 R)$ time in order to identify

[4] Heuristic approaches (Komodakis et al., 2007; Rush et al., 2010), tightening (Rush and Collins, 2011) or branch and bound (Das et al., 2012) can still be used to retrieve optimal solutions, but we did not explore these strategies here.

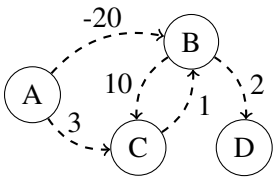

Figure 1: An example of the difficulty of recovering the maximum-weight subtree (B→C, B→D) from the maximum spanning tree (A→C, C→B, B→D).

the highest scoring sequence $\mathbf{y}$ and corresponding token configuration $\boldsymbol{\alpha}(\mathbf{y})$.

2.4 Dependency subtrees

The maximum-weight non-projective subtree problem over general graphs is not as easily solved. Although the maximum *spanning* tree for a given token configuration can be recovered efficiently, Figure 1 illustrates that the maximum-scoring subtree is not necessarily found within it. The problem of recovering a maximum-weight subtree in a graph has been shown to be NP-hard even with uniform edge weights (Lau et al., 2006).

In order to produce a solution to this subproblem, we use an LP relaxation of the relevant portion of the ILP from Thadani and McKeown (2013) by omitting integer constraints over the token and dependency variables in $\mathbf{x}$ and $\mathbf{z}$ respectively. For simplicity, however, we describe the ILP version rather than the relaxed LP in order to motivate the constraints with their intended purpose rather than their effect in the relaxed problem. The objective for this LP is given by

$$\max_{\mathbf{x},\mathbf{z}} \; \mathbf{x}^\top \boldsymbol{\theta'}_{\text{tok}} + \mathbf{z}^\top \boldsymbol{\theta}_{\text{dep}} \tag{5}$$

where the vector of token scores is redefined as

$$\boldsymbol{\theta'}_{\text{tok}} \triangleq (1 - \psi) \cdot \boldsymbol{\theta}_{\text{tok}} - \boldsymbol{\lambda} \tag{6}$$

in order to solve $g(\mathbf{z}, \boldsymbol{\lambda}, \psi, \boldsymbol{\theta})$ from (4).

Linear constraints are introduced to produce dependency structures that are close to the optimal dependency trees. First, tokens in the solution must only be active if they have a single active incoming dependency edge. In addition, to avoid producing multiple disconnected subtrees, only one dependency is permitted to attach to the ROOT pseudo-token.

$$x_j - \sum_i z_{ij} = 0, \quad \forall t_j \in T \tag{7}$$

$$\sum_j z_{ij} = 1, \quad \text{if } t_i = \text{ROOT} \tag{8}$$

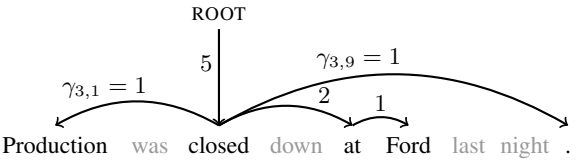

Figure 2: An illustration of commodity values for a valid solution of the non-relaxed ILP.

In order to avoid cycles in the dependency tree, we include additional variables to establish *single-commodity flow* (Magnanti and Wolsey, 1994) between all pairs of tokens. These γ_{ij} variables carry non-negative real values which must be consumed by active tokens that they are incident to.

$$\gamma_{ij} \geq 0, \quad \forall t_i, t_j \in T \tag{9}$$

$$\sum_i \gamma_{ij} - \sum_k \gamma_{jk} = x_j, \quad \forall t_j \in T \tag{10}$$

These constraints ensure that cyclic structures are not possible in the non-relaxed ILP. In addition, they serve to establish connectivity for the dependency structure $\mathbf{z}$ since commodity can only originate in one location—at the pseudo-token ROOT which has no incoming commodity variables. However, in order to enforce these properties on the output dependency structure, this acyclic, connected commodity structure must constrain the activation of the z variables.

$$\gamma_{ij} - C_{\text{max}} z_{ij} \leq 0, \quad \forall t_i, t_j \in T \tag{11}$$

where C_{max} is an arbitrary upper bound on the value of γ_{ij} variables. Figure 2 illustrates how these commodity flow variables constrain the output of the ILP to be a tree. However, the effect of these constraints is diminished when solving an LP relaxation of the above problem.

In the LP relaxation, x_i and z_{ij} are redefined as real-valued variables in $[0, 1]$, potentially resulting in fractional values for dependency and token indicators. As a result, the commodity flow network is able to establish connectivity but cannot enforce a tree structure, for instance, directed acyclic structures are possible and token indicators x_i may be partially be assigned to the solution structure. This poses a challenge in implementing $\boldsymbol{\beta}(\mathbf{z})$ which is needed to recover a token configuration from the solution of this subproblem.

We propose two alternative solutions to address this issue in the context of the joint inference strategy. The first is to simply use the relaxed token configuration identified by the LP in Algorithm 1,

i.e., to set $\boldsymbol{\beta}(\tilde{\mathbf{z}}) = \tilde{x}$ where $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{z}}$ represent the real-valued counterparts of the incidence vectors $\mathbf{x}$ and $\mathbf{z}$. The viability of this approximation strategy is due to the following:

- The relaxed LP is empirically fairly tight, yielding integral solutions 89% of the time on the compression datasets described in §3.
- The bigram subproblem is guaranteed to return a well-formed integral solution which obeys the imposed compression rate, so we are assured of a source of valid—if non-optimal—solutions in line 13 of Algorithm 1.

We also consider another strategy that attempts to approximate a valid integral solution to the dependency subproblem. In order to do this, we first include an additional constraint in the relaxed LP which restrict the number of tokens in the output to a specific number of tokens R that is given by an input compression rate.

$$\sum_i x_i = R \tag{12}$$

The addition of this constraint to the relaxed LP reduces the rate of integral solutions drastically—from 89% to approximately 33%—but it serves to ensure that the resulting token configuration $\tilde{\mathbf{x}}$ has at least as many non-zero elements as R, i.e., there are at least as many tokens activated in the LP solution as are required in a valid solution.

We then construct a subgraph $G(\tilde{\mathbf{z}})$ consisting of all dependency edges that were assigned non-zero values in the solution, assigning to each edge a score equal to the score of that edge in the LP as well as the score of its dependent word, i.e., each z_{ij} in $G(\tilde{\mathbf{z}})$ is assigned a score of $\theta_{\text{dep}}(\langle t_i, t_j \rangle) - \lambda_j + (1 - \psi) \cdot \theta_{\text{tok}}(t_j)$. Since the commodity flow constraints in (9)–(11) ensure a connected $\tilde{\mathbf{z}}$, it is therefore possible to recover a maximum-weight spanning tree from $G(\tilde{\mathbf{z}})$ using the Chu-Liu Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967).[5] Although the runtime of this algorithm is cubic in the size of the input graph, it is fairly speedy when applied on relatively sparse graphs such as the solutions to the LP described above. The resulting spanning tree is a useful integral approximation of $\tilde{\mathbf{z}}$ but, as mentioned previously, may contain more nodes than R due to fractional values in $\tilde{\mathbf{x}}$; we therefore repeatedly prune leaves with the lowest incoming edge weight in the current tree until exactly R nodes remain. The resulting tree is assumed to be a reasonable approximation of the optimal integral solution to this LP.

The Chu-Liu Edmonds algorithm is also employed for another purpose: when the underlying LP for the joint inference problem is not tight—a frequent occurrence in our compression experiments—Algorithm 1 will not converge on a single primal solution and will instead oscillate between solutions that are close to the dual optimum. We identify this phenomenon by counting repeated solutions and, if they exceed some threshold l_{max} with at least one repeated solution from either subproblem, we terminate the update procedure for Lagrange multipliers and instead attempt to identify a good solution from the repeating ones by scoring them under (2). It is straightforward to recover and score a bigram configuration $\mathbf{y}$ from a token configuration $\boldsymbol{\beta}(\mathbf{z})$. However, scoring solutions produced by the dynamic program from §2.3 also requires the score over a corresponding parse tree; this can be recovered by constructing a dependency subgraph containing across only the tokens that are active in $\boldsymbol{\alpha}(\mathbf{y})$ and retrieving the maximum spanning tree for that subgraph using the Chu-Liu Edmonds algorithm.

2.5 Learning and Features

The features used in this work are largely based on the features from Thadani and McKeown (2013).

- $\boldsymbol{\phi}_{\text{tok}}$ contains features for part-of-speech (POS) tag sequences of length up to 3 around the token, features for the dependency label of the token conjoined with its POS, lexical features for verb stems and non-word symbols and morphological features that identify capitalized sequences, negations and words in parentheses.
- $\boldsymbol{\phi}_{\text{bgr}}$ contains features for POS patterns in a bigram, the labels of dependency edges incident to it, its likelihood under a Gigaword language model (LM) and an indicator for whether it is present in the input sentence.
- $\boldsymbol{\phi}_{\text{dep}}$ contains features for the probability of a dependency edge under a smoothed dependency grammar constructed from the Penn Treebank and various conjunctions of the following features: (a) whether the edge appears as a dependency or ancestral relation in the input parse (b) the directionality of the depen-

[5] A detailed description of the Chu-Liu Edmonds algorithm for MSTs is available in McDonald et al. (2005).

dency (c) the label of the edge (d) the POS tags of the tokens incident to the edge and (e) the labels of their surrounding chunks and whether the edge remains within the chunk.

For the experiments in the following section, we trained models using a variant of the structured perceptron (Collins, 2002) which incorporates minibatches (Zhao and Huang, 2013) for easy parallelization and faster convergence.[6] Overfitting was avoided by averaging parameters and monitoring performance against a held-out development set during training. All models were trained using variants of the ILP-based inference approach of Thadani and McKeown (2013). We followed Martins et al. (2009) in using LP-relaxed inference during learning, assuming algorithmic separability (Kulesza and Pereira, 2007) for these problems.

3 Experiments

We ran compression experiments over the newswire (NW) and broadcast news transcription (BN) corpora compiled by Clarke and Lapata (2008) which contain gold compressions produced by human annotators using only word deletion. The datasets were filtered to eliminate instances with less than 2 and more than 110 tokens for parser compatibility and divided into training/development/test sections following the splits from Clarke and Lapata (2008), yielding 953/63/603 instances for the NW corpus and 880/78/404 for the BN corpus. Gold dependency parses were approximated by running the Stanford dependency parser[7] over reference compressions.

Following evaluations in machine translation as well as previous work in sentence compression (Unno et al., 2006; Clarke and Lapata, 2008; Martins and Smith, 2009; Napoles et al., 2011b; Thadani and McKeown, 2013), we evaluate system performance using F_1 metrics over n-grams and dependency edges produced by parsing system output with RASP (Briscoe et al., 2006) and the Stanford parser. All ILPs and LPs were solved using Gurobi,[8] a high-performance commercial-grade solver. Following a recent analysis of compression evaluations (Napoles et al., 2011b) which revealed a strong correlation between system compression rate and human judgments of compression quality, we constrained all systems to produce compressed output at a specific rate—determined by the the gold compressions available for each instance—to ensure that the reported differences between the systems under study are meaningful.

3.1 Systems

We report results over the following systems grouped into three categories of models: tokens + n-grams, tokens + dependencies, and joint models.

- **3-LM:** A reimplementation of the unsupervised ILP of Clarke and Lapata (2008) which infers order-preserving trigram variables parameterized with log-likelihood under an LM and a significance score for token variables inspired by Hori and Furui (2004), as well as various linguistically-motivated constraints to encourage fluency in output compressions.
- **DP:** The bigram-based dynamic program of McDonald (2006) described in §2.3.[9]
- **LP→MST:** An approximate inference approach based on an LP relaxation of **ILP-Dep**. As discussed in §2.4, a maximum spanning tree is recovered from the output of the LP and greedily pruned in order to generate a valid integral solution while observing the imposed compression rate.
- **ILP-Dep:** A version of the joint ILP of Thadani and McKeown (2013) without n-gram variables and corresponding features.
- **DP+LP→MST:** An approximate joint inference approach based on Lagrangian relaxation that uses **DP** for the maximum weight subsequence problem and **LP→MST** for the maximum weight subtree problem.
- **DP+LP:** Another Lagrangian relaxation approach that pairs **DP** with the non-integral solutions from an LP relaxation of the maximum weight subtree problem (cf. §2.4).
- **ILP-Joint:** The full ILP from Thadani and McKeown (2013), which provides an upper bound on the performance of the proposed approximation strategies.

The learning rate schedule for the Lagrangian relaxation approaches was set as $\eta_i \triangleq \tau/(\tau + i)$,[10] while the hyperparameter ψ was tuned using the

[6] We used a minibatch size of 4 in all experiments.

[7] `http://nlp.stanford.edu/software/`

[8] `http://www.gurobi.com`

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

[10] τ was set to 100 for aggressive subgradient updates.

Inference objective	Inference technique	n-grams $F_1\%$ $n=1$	2	3	4	Syntactic relations $F_1\%$ z	Stanford	RASP	Inference time (s)
n-grams	3-LM (CL08)	74.96	60.60	46.83	38.71	-	60.52	57.49	0.72
	DP (McD06)	78.80	66.04	52.67	42.39	-	63.28	57.89	0.01
deps	LP→MST	**79.61**	64.32	50.36	40.97	66.57	66.82	59.70	0.07
	ILP-Dep	**80.02**	65.99	52.42	43.07	**72.43**	67.63	60.78	0.16
joint	DP + LP→MST	79.50	66.75	53.48	44.33	64.63	67.69	60.94	0.24
	DP + LP	79.10	**68.22**	**55.05**	**45.81**	65.74	**68.24**	**62.04**	0.12
	ILP-Joint (TM13)	**80.13**	**68.34**	**55.56**	**46.60**	**72.57**	**68.89**	**62.61**	0.31

Table 1: Experimental results for the BN corpus, averaged over 3 gold compressions per instance. All systems were restricted to compress to the size of the median gold compression yielding an average compression rate of 77.26%.

Inference objective	Inference technique	n-grams $F_1\%$ $n=1$	2	3	4	Syntactic relations $F_1\%$ z	Stanford	RASP	Inference time (s)
n-grams	3-LM (CL08)	66.66	51.59	39.33	30.55	-	50.76	49.57	1.22
	DP (McD06)	73.18	58.31	45.07	34.77	-	56.23	51.14	0.01
deps	LP→MST	73.32	55.12	41.18	31.44	61.01	58.37	52.57	0.12
	ILP-Dep	**73.76**	57.09	43.47	33.44	**65.45**	60.06	54.31	0.28
joint	DP + LP→MST	73.13	57.03	43.79	34.01	57.91	58.46	53.20	0.33
	DP + LP	72.06	**59.83**	**47.39**	**37.72**	58.13	58.97	53.78	0.21
	ILP-Joint (TM13)	**74.00**	**59.90**	**47.22**	**37.01**	**65.65**	**61.29**	**56.24**	0.60

Table 2: Experimental results for the NW corpus with all systems compressing to the size of the gold compression, yielding an average compression rate of 70.24%. In both tables, bold entries show significant gains within a column under the paired t-test ($p < 0.05$) and Wilcoxon's signed rank test ($p < 0.01$).

development split of each corpus.[11]

3.2 Results

Tables 1 and 2 summarize the results from our compression experiments on the BN and NW corpora respectively. Starting with the n-gram approaches, the performance of **3-LM** leads us to observe that the gains of supervised learning far outweigh the utility of higher-order n-gram factorization, which is also responsible for a significant increase in wall-clock time. In contrast, **DP** is an order of magnitude faster than all other approaches studied here although it is not competitive under parse-based measures such as RASP $F_1\%$ which is known to correlate with human judgments of grammaticality (Clarke and Lapata, 2006).

We were surprised by the strong performance of the dependency-based inference techniques, which yielded results that approached the joint model in both n-gram and parse-based measures. The exact **ILP-Dep** approach halves the runtime of **ILP-Joint** to produce compressions that have similar (although statistically distinguishable) scores. Approximating dependency-based inference with **LP→MST** yields similar performance for a further halving of runtime; however, the performance of this approach is notably worse.

Turning to the joint approaches, the strong performance of **ILP-Joint** is expected; less so is the relatively high but yet practically reasonable runtime that it requires. We note, however, that these ILPs are solved using a highly-optimized commercial-grade solver that can utilize all CPU cores[12] while our approximation approaches are implemented as single-processed Python code without significant effort toward optimization. Comparing the two approximation strategies shows a clear performance advantage for **DP+LP** over **DP+LP→MST**: the latter approach entails slower inference due to the overhead of running the Chu-Liu Edmonds algorithm at every dual update, and furthermore, the error introduced by approximating an integral solution re-

[11] We were surprised to observe that performance improved significantly when ψ was set closer to 1, thereby emphasizing token features in the dependency subproblem. The final values chosen were $\psi_{\text{BN}} = 0.9$ and $\psi_{\text{NW}} = 0.8$.

[12] 16 cores in our experimental environment.

sults in a significant decrease in dependency recall. In contrast, **DP+LP** directly optimizes the dual problem by using the relaxed dependency solution to update Lagrange multipliers and achieves the best performance on parse-based F_1 outside of the slower ILP approaches. Convergence rates also vary for these two techniques: **DP+LP** has a lower rate of empirical convergence (15% on BN and 4% on NW) when compared to **DP+LP→MST** (19% on BN and 6% on NW).

Figure 3 shows the effect of input sentence length on inference time and performance for **ILP-Joint** and **DP+LP** over the NW test corpus.[13] The timing results reveal that the approximation strategy is consistently faster than the ILP solver. The variation in RASP F_1% with input size indicates the viability of a hybrid approach which could balance accuracy and speed by using **ILP-Joint** for smaller problems and **DP+LP** for larger ones.

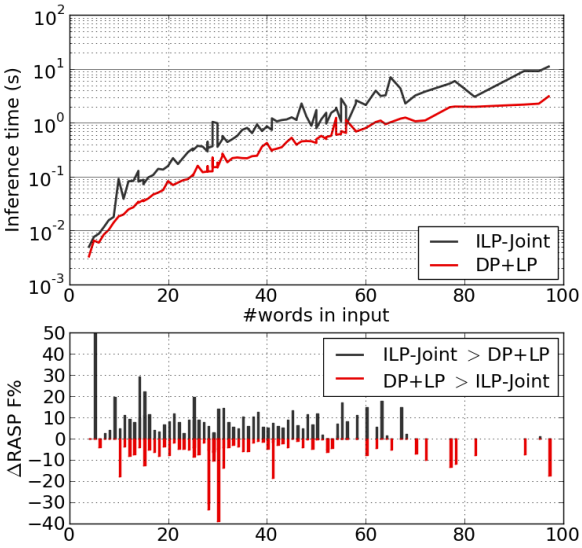

Figure 3: Effect of input size on (a) inference time, and (b) the corresponding difference in RASP F_1% (**ILP-Joint** – **DP+LP**) on the NW corpus.

4 Related Work

Sentence compression is one of the better-studied text-to-text generation problems and has been observed to play a significant role in human summarization (Jing, 2000; Jing and McKeown, 2000). Most approaches to sentence compression are supervised (Knight and Marcu, 2002; Riezler et al., 2003; Turner and Charniak, 2005; McDonald, 2006; Unno et al., 2006; Galley and McKeown, 2007; Nomoto, 2007; Cohn and Lapata, 2009; Galanis and Androutsopoulos, 2010; Ganitkevitch et al., 2011; Napoles et al., 2011a; Filippova and Altun, 2013) following the release of datasets such as the Ziff-Davis corpus (Knight and Marcu, 2000) and the Edinburgh compression corpora (Clarke and Lapata, 2006; Clarke and Lapata, 2008), although unsupervised approaches—largely based on ILPs—have also received consideration (Clarke and Lapata, 2007; Clarke and Lapata, 2008; Filippova and Strube, 2008). Compression has also been used as a tool for document summarization (Daumé and Marcu, 2002; Zajic et al., 2007; Clarke and Lapata, 2007; Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Woodsend and Lapata, 2012; Almeida and Martins, 2013; Molina et al., 2013; Li et al., 2013; Qian and Liu, 2013), with recent work formulating the summarization task as joint sentence extraction and compression and often employing ILP or Lagrangian relaxation. Monolingual compression also faces many obstacles common to decoding in machine translation, and a number of approaches which have been proposed to combine phrasal and syntactic models (Huang and Chiang, 2007; Rush and Collins, 2011) *inter alia* offer directions for future research into compression problems.

5 Conclusion

We have presented approximate inference strategies to jointly compress sentences under bigram and dependency-factored objectives by exploiting the modularity of the task and considering the two subproblems in isolation. Experiments show that one of these approximation strategies produces results comparable to a state-of-the-art integer linear program for the same joint inference task with a 60% reduction in average inference time.

Acknowledgments

The author is grateful to Alexander Rush for helpful discussions and to the anonymous reviewers for their comments. This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20153. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.[14]

[13] Similar results were observed for the BN test corpus.

[14] The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

Miguel Almeida and André F. T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of ACL*, pages 196–206, August.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of ACL-HLT*, pages 481–490.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the ACL-COLING Interactive Presentation Sessions*.

Yoeng-jin Chu and Tseng-hong Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: a comparison across domains, training requirements and evaluation measures. In *Proceedings of ACL-COLING*, pages 377–384.

James Clarke and Mirella Lapata. 2007. Modelling compression with discourse constraints. In *Proceedings of EMNLP-CoNLL*, pages 1–11.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: an integer linear programming approach. *Journal for Artificial Intelligence Research*, 31:399–429, March.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of COLING*, pages 137–144.

Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34(1):637–674, April.

Michael Collins. 2002. Discriminative training methods for hidden Markov models. In *Proceedings of EMNLP*, pages 1–8.

Dipanjan Das, André F. T. Martins, and Noah A. Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, SemEval '12, pages 209–217.

Hal Daumé, III and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of ACL*, pages 449–456.

John DeNero and Klaus Macherey. 2011. Model-based aligner combination using dual decomposition. In *Proceedings of ACL-HLT*, pages 420–429.

Jack R. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of EMNLP*, pages 1481–1491.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of INLG*, pages 25–32.

Dimitrios Galanis and Ion Androutsopoulos. 2010. An extractive supervised two-stage method for sentence compression. In *Proceedings of HLT-NAACL*, pages 885–893.

Michel Galley and Kathleen McKeown. 2007. Lexicalized Markov grammars for sentence compression. In *Proceedings of HLT-NAACL*, pages 180–187, April.

Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and Benjamin Van Durme. 2011. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of EMNLP*, pages 1168–1179.

Chiori Hori and Sadaoki Furui. 2004. Speech summarization: an approach through word extraction and a method for evaluation. *IEICE Transactions on Information and Systems*, E87-D(1):15–25.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL*, pages 144–151, June.

Hongyan Jing and Kathleen R. McKeown. 2000. Cut and paste based text summarization. In *Proceedings of NAACL*, pages 178–185.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the Conference on Applied Natural Language Processing*, pages 310–315.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of AAAI*, pages 703–710.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107, July.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *Proceedings of ICCV*, pages 1–8, Oct.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP*, pages 1288–1298.

Alex Kulesza and Fernando Pereira. 2007. Structured learning with approximate inference. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. Curran Associates, Inc.

Hoong Chuin Lau, Trung Hieu Ngo, and Bao Nguyen Nguyen. 2006. Finding a length-constrained maximum-sum or maximum-density subtree and its application to logistics. *Discrete Optimization*, 3(4):385 – 391.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *Proceedings of EMNLP*, pages 490–500, Seattle, Washington, USA, October.

Thomas L. Magnanti and Laurence A. Wolsey. 1994. Optimal trees. In *Technical Report 290-94, Massechusetts Institute of Technology, Operations Research Center*.

André F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 1–9.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL-IJCNLP*, pages 342–350.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proceedings of EMNLP*, pages 238–249.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of EMNLP-HLT*, pages 523–530.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of EACL*, pages 297–304.

Alejandro Molina, Juan-Manuel Torres-Moreno, Eric SanJuan, Iria da Cunha, and Gerardo Eugenio Sierra Martínez. 2013. Discursive sentence compression. In *Computational Linguistics and Intelligent Text Processing*, volume 7817, pages 394–407. Springer.

Courtney Napoles, Chris Callison-Burch, Juri Ganitkevitch, and Benjamin Van Durme. 2011a. Paraphrastic sentence compression with a character-based metric: tightening without deletion. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 84–90.

Courtney Napoles, Benjamin Van Durme, and Chris Callison-Burch. 2011b. Evaluating sentence compression: pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 91–97.

Tadashi Nomoto. 2007. Discriminative sentence compression with conditional random fields. *Information Processing and Management*, 43(6):1571–1587, November.

Xian Qian and Yang Liu. 2013. Fast joint compression and summarization via graph cuts. In *Proceedings of EMNLP*, pages 1492–1502, Seattle, Washington, USA, October.

Stefan Riezler, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of HLT-NAACL*, pages 118–125.

Alexander M. Rush and Michael Collins. 2011. Exact decoding of syntactic translation models through Lagrangian relaxation. In *Proceedings of ACL-HLT*, pages 72–82.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of EMNLP*, pages 1–11.

Kapil Thadani and Kathleen McKeown. 2013. Sentence compression with joint structural inference. In *Proceedings of CoNLL*.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of ACL*, pages 290–297.

Yuya Unno, Takashi Ninomiya, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Trimming CFG parse trees for sentence compression using machine learning approaches. In *Proceedings of ACL-COLING*, pages 850–857.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of EMNLP*, pages 233–243.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing and Management*, 43(6):1549–1570, Nov.

Kai Zhao and Liang Huang. 2013. Minibatch and parallelization for online large margin structured learning. In *Proceedings of HLT-NAACL*, pages 370–379, Atlanta, Georgia, June.