Journal of Artificial Intelligence Research 31 (2008) 399-429 Submitted 09/07; published 03/08

Global Inference for Sentence Compression An Integer Linear Programming Approach

James Clarke JCLARKE@ED.AC.UK

Mirella Lapata MLAP@INF.ED.AC.UK

School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, UK

Abstract

Sentence compression holds promise for many applications ranging from summarization to subtitle generation. Our work views sentence compression as an optimization problem and uses integer linear programming (ILP) to infer globally optimal compressions in the presence of linguistically motivated constraints. We show how previous formulations of sentence compression can be recast as ILPs and extend these models with novel global constraints. Experimental results on written and spoken texts demonstrate improvements over state-of-the-art models.

1. Introduction

The computational treatment of sentence compression has recently attracted much attention in the literature. The task can be viewed as producing a summary of a single sentence that retains the most important information and remains grammatical (Jing, 2000). A sentence compression mechanism would greatly benefit a wide range of applications. For example, in summarization, it could improve the conciseness of the generated summaries (Jing, 2000; Lin, 2003; Zajic, Door, Lin, & Schwartz, 2007). Other examples include compressing text to be displayed on small screens such as mobile phones or PDAs (Corston-Oliver, 2001), subtitle generation from spoken transcripts (Vandeghinste & Pan, 2004), and producing audio scanning devices for the blind (Grefenstette, 1998).

Sentence compression is commonly expressed as a word deletion problem: given an input *source* sentence of words $\mathbf{x} = x_1, x_2, \ldots, x_n$, the aim is to produce a *target* compression by removing any subset of these words (Knight & Marcu, 2002). The compression problem has been extensively studied across different modeling paradigms, both supervised and unsupervised. Supervised models are typically trained on a parallel corpus of source sentences and target compressions and come in many flavors. Generative models aim to model the probability of a target compression given the source sentence either directly (Galley & McKeown, 2007) or indirectly using the noisy-channel model (Knight & Marcu, 2002; Turner & Charniak, 2005), whereas discriminative formulations attempt to minimize error rate on a training set. These include decision-tree learning (Knight & Marcu, 2002), maximum entropy (Riezler, King, Crouch, & Zaenen, 2003), support vector machines (Nguyen, Shimazu, Horiguchi, Ho, & Fukushi, 2004), and large-margin learning (McDonald, 2006).

Unsupervised methods dispense with the parallel corpus and generate compressions either using rules (Turner & Charniak, 2005) or a language model (Hori & Furui, 2004).

Despite differences in formulation, all these approaches model the compression process using *local* information. For instance, in order to decide which words to drop, they exploit information about adjacent words or constituents. Local models can do a good job at producing grammatical compressions, however they are somewhat limited in scope since they cannot incorporate *global* constraints on the compression output. Such constraints consider the sentence as a whole instead of isolated linguistic units (words or constituents). To give a concrete example we may want to ensure that each target compression has a verb, provided that the source had one in the first place. Or that verbal arguments are present in the compression. Or that pronouns are retained. Such constraints are fairly intuitive and can be used to instill not only linguistic but also task specific information into the model. For instance, an application which compresses text to be displayed on small screens would presumably have a higher compression rate than a system generating subtitles from spoken text. A global constraint could force the former system to generate compressions with a fixed rate or a fixed number of words.

Existing approaches do not model global properties of the compression problem for a good reason. Finding the best compression for a source sentence given the space of all possible compressions[1] (this search process is often referred to as decoding or inference) can become intractable for too many constraints and overly long sentences. Typically, the decoding problem is solved efficiently using dynamic programming often in conjunction with heuristics that reduce the search space (e.g., Turner & Charniak, 2005). Dynamic programming guarantees we will find the global optimum provided the principle of optimality holds. This principle states that given the current state, the optimal decision for each of the remaining stages does not depend on previously reached stages or previously made decisions (Winston & Venkataramanan, 2003). However, we know this to be false in the case of sentence compression. For example, if we have included modifiers to the left of a noun in a compression then we should probably include the noun too or if we include a verb we should also include its arguments. With a dynamic programming approach we cannot easily guarantee such constraints hold.

In this paper we propose a novel framework for sentence compression that incorporates constraints on the compression output and allows us to find an optimal solution. Our formulation uses integer linear programming (ILP), a general-purpose exact framework for NP-hard problems. Specifically, we show how previously proposed models can be recast as integer linear programs. We extend these models with constraints which we express as linear inequalities. Decoding in this framework amounts to finding the best solution given a linear (scoring) function and a set of linear constraints that can be either global or local. Although ILP has been previously used for sequence labeling tasks (Roth & Yih, 2004; Punyakanok, Roth, Yih, & Zimak, 2004), its application to natural language generation is less widespread. We present three compression models within the ILP framework, each representative of an unsupervised (Knight & Marcu, 2002), semi-supervised (Hori & Furui, 2004), and fully supervised modeling approach (McDonald, 2006). We propose a small number of constraints ensuring that the compressions are structurally and semantically

[1] There are 2^n possible compressions where n is the number of words in a sentence.

valid and experimentally evaluate their impact on the compression task. In all cases, we show that the added constraints yield performance improvements.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work. In Section 3 we present the ILP framework and the compression models we employ in our experiments. Our constraints are introduced in Section 3.5. Section 4.3 discusses our experimental set-up and Section 5 presents our results. Discussion of future work concludes the paper.

2. Related Work

In this paper we develop several ILP-based compression models. Before presenting these models, we briefly summarize previous work addressing sentence compression with an emphasis on data-driven approaches. Next, we describe how ILP techniques have been used in the past to solve other inference problems in natural language processing (NLP).

2.1 Sentence Compression

Jing (2000) was perhaps the first to tackle the sentence compression problem. Her approach uses multiple knowledge sources to determine which phrases in a sentence to remove. Central to her system is a grammar checking module that specifies which sentential constituents are grammatically obligatory and should therefore be present in the compression. This is achieved using simple rules and a large-scale lexicon. Other knowledge sources include WordNet and corpus evidence gathered from a parallel corpus of source-target sentence pairs. A phrase is removed only if it is not grammatically obligatory, not the focus of the local context and has a reasonable deletion probability (estimated from a parallel corpus).

In contrast to Jing (2000), the bulk of the research on sentence compression relies exclusively on corpus data for modeling the compression process without recourse to extensive knowledge sources (e.g., WordNet). A large number of approaches are based on the noisy-channel model (Knight & Marcu, 2002). These approaches consist of a language model $P(y)$ (whose role is to guarantee that compression output is grammatical), a channel model $P(x|y)$ (capturing the probability that the source sentence x is an expansion of the target compression y), and a decoder (which searches for the compression y that maximizes $P(y)P(x|y)$). The channel model is acquired from a parsed version of a parallel corpus; it is essentially a stochastic synchronous context-free grammar (Aho & Ullman, 1969) whose rule probabilities are estimated using maximum likelihood. Modifications of this model are presented by Turner and Charniak (2005) and Galley and McKeown (2007) with improved results.

In discriminative models (Knight & Marcu, 2002; Riezler et al., 2003; McDonald, 2006; Nguyen et al., 2004) sentences are represented by a rich feature space (also induced from parse trees) and the goal is to learn which words or word spans should be deleted in a given context. For instance, in Knight and Marcu's (2002) decision-tree model, compression is performed deterministically through a tree rewriting process inspired by the shift-reduce parsing paradigm. Nguyen et al. (2004) render this model probabilistic through the use of support vector machines. McDonald (2006) formalizes sentence compression in a large-margin learning framework without making reference to shift-reduce parsing. In his model compression is a classification task: pairs of words from the source sentence are classified as being adjacent or not in the target compression. A large number of features are defined over words, parts-of-speech, phrase structure trees and dependencies. These features are gathered over adjacent words in the compression and the words in-between which were dropped (see Section 3.4.3 for a more detailed account).

While most compression models have been developed with written text in mind, Hori and Furui (2004) propose a model for automatically transcribed spoken text. Their model generates compressions through word deletion without using parallel data or syntactic information in any way. Assuming a fixed compression rate, it searches for the compression with the highest score using a dynamic programming algorithm. The scoring function consists of a language model responsible for producing grammatical output, a significance score indicating whether a word is topical or not, and a score representing the speech recognizer's confidence in transcribing a given word correctly.

2.2 Integer Linear Programming in NLP

ILPs are constrained optimization problems where both the objective function and the constraints are linear equations with integer variables (see Section 3.1 for more details). ILP techniques have been recently applied to several NLP tasks, including relation extraction (Roth & Yih, 2004), semantic role labeling (Punyakanok et al., 2004), the generation of route directions (Marciniak & Strube, 2005), temporal link analysis (Bramsen, Deshpande, Lee, & Barzilay, 2006), set partitioning (Barzilay & Lapata, 2006), syntactic parsing (Riedel & Clarke, 2006), and coreference resolution (Denis & Baldridge, 2007).

Most of these approaches combine a local classifier with an inference procedure based on ILP. The classifier proposes possible answers which are assessed in the presence of global constraints. ILP is used to make a final decision that is consistent with the constraints and likely according to the classifier. For example, the semantic role labeling task involves identifying the verb-argument structure for a given sentence. Punyakanok et al. (2004) first use SNOW, a multi-class classifier[2] (Roth, 1998), to identify and label candidate arguments. They observe that the labels assigned to arguments in a sentence often contradict each other. To resolve these conflicts they propose global constraints (e.g., each argument should be instantiated once for a given verb, every verb should have at least one argument) and use ILP to reclassify the output of SNOW.

Dras (1999) develops a document paraphrasing model using ILP. The key premise of his work is that in some cases one may want to rewrite a document so as to conform to some global constraints such as length, readability, or style. The proposed model has three ingredients: a set of sentence-level paraphrases for rewriting the text, a set of global constraints, and an objective function which quantifies the effect incurred by the paraphrases. Under this formulation, ILP can be used to select which paraphrases to apply so that the global constraints are satisfied. Paraphrase generation falls outside the scope of the ILP model – sentence rewrite operations are mainly syntactic and provided by a module based on synchronous tree adjoining grammar (S-TAG, Shieber & Schabes, 1990). Unfortunately, only a proof-of-concept is presented; implementation and evaluation of this module are left to future work.

[2] SNOW's learning algorithm is a variation of the Winnow update rule.

Our work models sentence compression as an optimization problem. We show how previously proposed models can be reformulated in the context of integer linear programming which allows us to easily incorporate constraints during the decoding process. Our constraints are linguistically and semantically motivated and are designed to bring less local syntactic knowledge into the model and help preserve the meaning of the source sentence. Previous work has identified several important features for the compression task (Knight & Marcu, 2002; McDonald, 2006); however, the use of global constraints is novel to our knowledge. Although sentence compression has not been explicitly formulated in terms of optimization, previous approaches rely on some optimization procedure for generating the best compression. The decoding process in the noisy-channel model searches for the best compression given the source and channel models. However, the compression found is usually sub-optimal as heuristics are used to reduce the search space or is only locally optimal due to the search method employed. For example, in the work of Turner and Charniak (2005) the decoder first searches for the best combination of rules to apply. As it traverses the list of compression rules, it removes sentences outside the 100 best compressions (according to the channel model). This list is eventually truncated to 25 compressions. In other models (Hori & Furui, 2004; McDonald, 2006) the compression score is maximized using dynamic programming which however can yield suboptimal results (see the discussion in Section 1).

Contrary to most other NLP work using ILP (a notable exception is Roth & Yih, 2005), we do not view compression generation as a two stage process where learning and inference are carried out sequentially (i.e., first a local classifier hypothesizes a list of possible answers and then the best answer is selected using global constraints). Our models integrate learning with inference in a unified framework where decoding takes place in the presence of all available constraints, both local and global. Moreover, we investigate the influence of our constraint set across models and learning paradigms. Previous work typically formulates constraints for a single model (e.g., the SNOW classifier) and learning paradigm (e.g., supervised). We therefore assess how the constraint-based framework advocated in this article influences the performance of expressive models (which require large amounts of parallel data) and non-expressive ones (which use very little parallel data or none at all). In other words, we are able to pose and answer the following question: what kinds of models benefit most from constraint-based inference?

Our work is close in spirit but rather different in content to Dras (1999). We concentrate on compression, a specific paraphrase type, and apply our models on the sentence-level. Our constraints thus do not affect the document as a whole but individual sentences. Furthermore, compression generation is an integral part of our ILP models, whereas Dras assumes that paraphrases are generated by a separate process.

3. Framework

In this section we present the details of the proposed framework for sentence compression. As mentioned earlier, our work models sentence compression directly as an optimization problem. There are 2^n possible compressions for each source sentence and while many of these will be unreasonable, it is unlikely that only one compression will be satisfactory (Knight & Marcu, 2002). Ideally, we require a function that captures the operations (or rules) that can be performed on a sentence to create a compression while at the same time factoring how desirable each operation makes the resulting compression. We can then perform a search over *all* possible compressions and select the best one, as determined by how desirable it is. A wide range of models can be expressed under this framework. The prerequisites for implementing these are fairly low, we only require that the decoding process be expressed as a linear function with a set of linear constraints. In practice, many models rely on a Markov assumption for factorization which is usually solved with a dynamic programming-based decoding process. Such algorithms can be formulated as integer linear programs with little effort.

We first give a brief introduction into integer linear programming, an extension of linear programming for readers unfamiliar with mathematical programming. Our compression models are next described in Section 3.4 and constraints in Section 3.5.

3.1 Linear Programming

Linear programming (LP) problems are optimization problems with constraints. They consist of three parts:

- Decision variables. These are variables under our control which we wish to assign optimal values to.
- A linear function (the *objective function*). This is the function we wish to minimize or maximize. This function is influences by the values assigned to the decision variables.
- Constraints. Most problems will only allow the decision variables to take certain values. These restrictions are the constraints.

These terms are best demonstrated with a simple example taken from Winston and Venkataramanan (2003). Imagine a manufacturer of tables and chairs which we shall call the Telfa Corporation. To produce a table, 1 hour of labor and 9 square board feet of wood is required. Chairs require 1 hour of labor and 5 square board feet of wood. Telfa have 6 hours of labor and 45 square board feet of wood available. The profit made from each table is 8 GBP and 5 GBP for chairs. We wish to determine the number of tables and chairs that should be manufactured to maximize Telfa's profit.

First, we must determine the *decision variables.* In our case we define:

$$\begin{aligned} x_1 &= \text{number of tables manufactured} \\ x_2 &= \text{number of chairs manufactured} \end{aligned}$$

Our objective function is the value we wish to maximize, namely the profit.

$$\text{Profit} = 8x_1 + 5x_2$$

There are two constraints in this problem: we must not exceed 6 hours of labor and no more than 45 square board feet of wood must be used. Also, we cannot create a negative amount of chairs or tables:

$$\begin{array}{lrcrcr}
\text{Labor constraint} & x_1 & + & x_2 & \leq & 6 \\
\text{Wood constraint} & 9x_1 & + & 5x_2 & \leq & 45 \\
\text{Variable constraints} & & & x_1 & \geq & 0 \\
 & & & x_2 & \geq & 0
\end{array}$$

Once the decision variables, objective function and constraints have been determined we can express the LP model:

$$\max z = 8x_1 + 5x_2 \quad \text{(Objective function)}$$

subject to (s.t.)

$$\begin{array}{rcrcrl}
x_1 & + & x_2 & \leq & 6 & \text{(Labor constraint)} \\
9x_1 & + & 5x_2 & \leq & 45 & \text{(Wood constraint)} \\
 & & x_1 & \geq & 0 & \\
 & & x_2 & \geq & 0 &
\end{array}$$

Two of the most basic concepts involved in solving LP problems are the *feasibility region* and *optimal solution.* The optimal solution is one in which all constraints are satisfied and the objective function is minimized or maximized. A specification of the value for each decision variable is referred to as a *point.* The feasibility region for a LP is a region consisting of the set of all points that satisfy all the LP's constraints. The optimal solution lies within this feasibility region, it is the point with the minimum or maximum objective function value.

A set of points satisfying a single linear inequality is a *half-space.* The feasibility region is defined by a the intersection of m half-spaces (for m linear inequalities) and forms a *polyhedron.* Our Telfa example forms a polyhedral set (a polyhedral convex set) from the intersection of our four constraints. Figure 1a shows the feasible region for the Telfa example. To find the optimal solution we graph a line (or hyperplane) on which all points have the same objective function value. In maximization problems it is called the *isoprofit line* and in minimization problems the *isocost line.* One isoprofit line is represented by the dashed black line in Figure 1a. Once we have one isoprofit line we can find all other isoprofit lines by moving parallel to the original isoprofit line.

The *extreme points* of the polyhedral set are defined as the intersections of the lines that form the boundaries of the polyhedral set (points A B C and D in Figure 1a). It can be shown that any LP that has an optimal solution, has an extreme point that is globally optimal. This reduces the search space of the optimization problem to finding the extreme point with the highest or lowest value. The simplex algorithm (Dantzig, 1963) solves LPs by exploring the extreme points of a polyhedral set. Specifically, it moves from one extreme point to an *adjacent extreme point* (extreme points that lie on the same line segment) until an optimal extreme point is found. Although the simplex algorithm has an exponential worst-case complexity, in practice the algorithm is very efficient.

The optimal solution for the Telfa example is $z = \frac{165}{4}$, $x_1 = \frac{15}{4}$, $x_2 = \frac{9}{4}$. Thus, to achieve a maximum profit of 41.25 GBP they must build 3.75 tables and 2.25 chairs. This is obviously impossible as we would not expect people to buy fractions of tables and chairs. Here, we want to be able to constrain the problem such that the decision variables can only take integer values. This can be done with Integer Linear Programming.

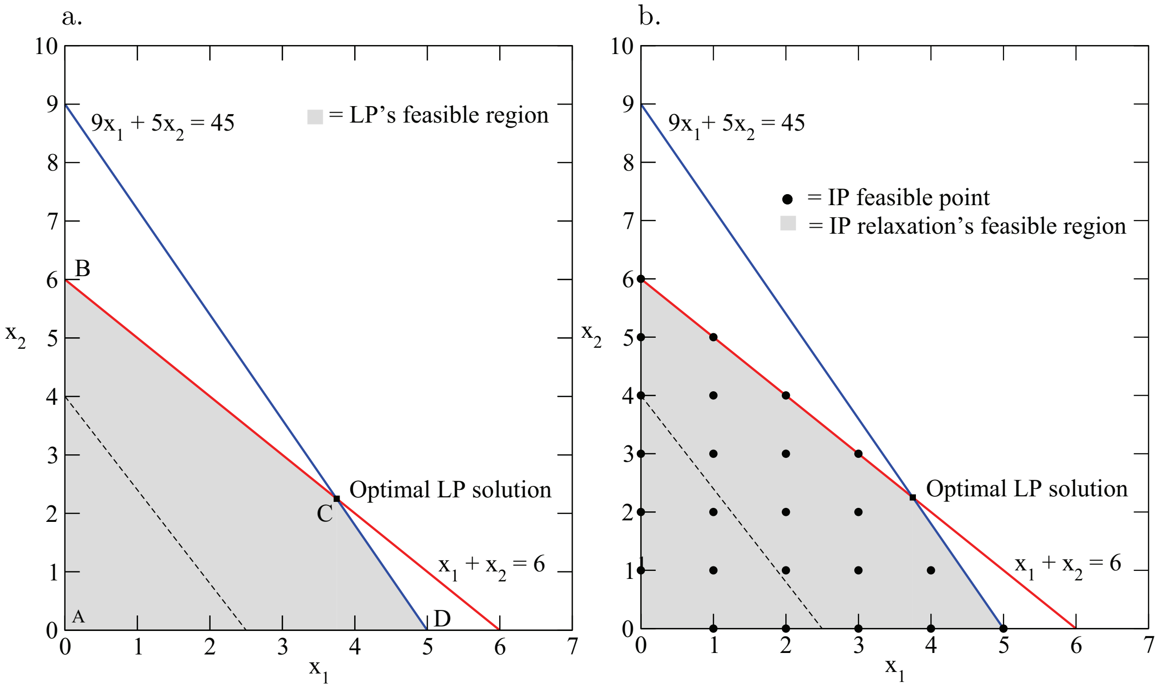

Figure 1: Feasible region for the Telfa example using linear (graph (a)) and integer linear (graph (b)) programming

3.2 Integer Linear Programming

Integer linear programming (ILP) problems are LP problems in which some or all of the variables are required to be non-negative integers. They are formulated in a similar manner to LP problems with the added constraint that all decision variables must take non-negative integer values.

To formulate the Telfa problem as an ILP model we merely add the constraints that x_1 and x_2 must be integer. This gives:

$$\max z = 8x_1 + 5x_2 \quad \text{(Objective function)}$$

subject to (s.t.)

$$\begin{array}{rcrcrl} x_1 & + & x_2 & \leq & 6 & \text{(Labor constraint)} \\ 9x_1 & + & 5x_2 & \leq & 45 & \text{(Wood constraint)} \\ & & x_1 & \geq & 0; x_1 \text{ integer} & \\ & & x_2 & \geq & 0; x_2 \text{ integer} & \end{array}$$

For LP models, it can be proved that the optimal solution lies on an extreme point of the feasible region. In the case of integer linear programs, we only wish to consider points that are integer values. This is illustrated in Figure 1b for the Telfa problem. In contrast to linear programming, which can be solved efficiently in the worst case, integer programming problems are in many practical situations NP-hard (Cormen, Leiserson, & Rivest, 1992).

Fortunately, ILPs are a well studied optimization problem and a number of techniques have been developed to find the optimal solution. Two such techniques are the cutting planes method (Gomory, 1960) and the branch-and-bound method (Land & Doig, 1960). We briefly discuss these methods here. For a more detailed treatment we refer the interested reader to Winston and Venkataramanan (2003) or Nemhauser and Wolsey (1988).

The cutting planes method adds extra constraints to slice parts of the feasible region until it contains only integer extreme points. However, this process can be difficult or impossible (Nemhauser & Wolsey, 1988). The branch-and-bound method enumerates all points in the ILP's feasible region but prunes those sections in the region which are known to be sub-optimal. It does this by *relaxing* the integer constraints and solving the resulting LP problem (known as the *LP relaxation*). If the solution of the LP relaxation is integral, then it is the optimal solution. Otherwise, the resulting solution provides an upper bound on the solution for the ILP. The algorithm proceeds by creating two new sub-problems based on the non-integer solution for one variable at a time. These are solved and the process repeats until the optimal integer solution is found.

Using the branch-and-bound method, we find that the optimal solution to the Telfa problem is $z = 40, x_1 = 5, x_2 = 0$; thus, to achieve a maximum profit of 40 GBP, Telfa must manufacture 5 tables and 0 chairs. This is a relatively simple problem, which could be solved merely by inspection. Most ILP problems will involve many variables and constraints resulting in a feasible region with a large number of integer points. The branch-and-bound procedure can efficiently solve such ILPs in a matter of seconds and forms part of many commercial ILP solvers. In our experiments we use *lp_solve*[3], a free optimization package which relies on the simplex algorithm and brand-and-bound methods for solving ILPs.

Note that under special circumstances other solving methods may be applicable. For example, implicit enumeration can be used to solve ILPs where all the variables are binary (also known as pure $0-1$ problems). Implicit enumeration is similar to the branch-and-bound method, it systematically evaluates all possible solutions, without however explicitly solving a (potentially) large number of LPs derived from the relaxation. This removes much of the computational complexity involved in determining if a sub-problem is infeasible. Furthermore, for a class of ILP problems known as minimum cost network flow problems (MCNFP), the LP relaxation always yields an integral solution. These problems can therefore be treated as LP problems.

In general, a model will yield an optimal solution in which all variables are integers if the constraint matrix has a property known as *total unimodularity.* A matrix A is totally unimodular if every square sub-matrix of A has its determinant equal to 0, $+1$ or -1. It is the case that the more the constraint matrix looks totally unimodular, the easier the problem will be to solve by branch-and-bound methods. In practice it is good to formulate ILPs where as many variables as possible have coefficients of 0, $+1$ or -1 in the constraints (Winston & Venkataramanan, 2003).

3.3 Constraints and Logical Conditions

Although integer variables in ILP problems may take arbitrary values, these are frequently are restricted to 0 and 1. Binary variables ($0-1$ variables) are particularly useful for rep-

[3] The software is available from `http://lpsolve.sourceforge.net/`.

Condition	Statement	Constraint
Implication	if a then b	$b - a \geq 0$
Iff	a if and only if b	$a - b = 0$
Or	a or b or c	$a + b + c \geq 1$
Xor	a xor b xor c	$a + b + c = 1$
And	a and b	$a = 1; b = 1$
Not	not a	$1 - a = 1$

Table 1: How to represent logical conditions using binary variables and constraints in ILP.

resenting a variety of logical conditions within the ILP framework through the use of constraints. Table 1 lists several logical conditions and their equivalent constraints.

We can also express transitivity, i.e., "c if and only if a and b". Although it is often thought that transitivity can only be expressed as a polynomial expression of binary variables (i.e., $ab = c$), it is possible to replace the latter by the following linear inequalities (Williams, 1999):

$$\begin{aligned}(1-c)+a &\geq 1\\(1-c)+b &\geq 1\\c+(1-a)+(1-b) &\geq 1\end{aligned}$$

This can be easily extended to model indicator variables representing whether a set of binary variables can take certain values.

3.4 Compression Models

In this section we describe three compression models which we reformulate as integer linear programs. Our first model is a simple language model which has been used as a baseline in previous research (Knight & Marcu, 2002). Our second model is based on the work of Hori and Furui (2004); it combines a language model with a corpus-based significance scoring function (we omit here the confidence score derived from the speech recognizer since our models are applied to text only). This model requires a small amount of parallel data to learn weights for the language model and the significance score.

Our third model is fully supervised, it uses a discriminative large-margin framework (McDonald, 2006), and is trained trained on a larger parallel corpus. We chose this model instead of the more popular noisy-channel or decision-tree models, for two reasons, a practical one and a theoretical one. First, McDonald's (2006) model delivers performance superior to the decision-tree model (which in turn performs comparably to the noisy-channel). Second, the noisy channel is not an entirely appropriate model for sentence compression. It uses a language model trained on uncompressed sentences even though it represents the probability of compressed sentences. As a result, the model will consider compressed sentences less likely than uncompressed ones (a further discussion is provided by Turner & Charniak, 2005).

3.4.1 Language Model

A language model is perhaps the simplest model that springs to mind. It does not require a parallel corpus (although a relatively large monolingual corpus is necessary for training), and will naturally prefer short sentences to longer ones. Furthermore, a language model can be used to drop words that are either infrequent or unseen in the training corpus. Knight and Marcu (2002) use a bigram language model as a baseline against their noisy-channel and decision-tree models.

Let $\mathbf{x} = x_1, x_2, \ldots, x_n$ denote a source sentence for which we wish to generate a target compression. We introduce a decision variable for each word in the source and constrain it to be binary; a value of 0 represents a word being dropped, whereas a value of 1 includes the word in the target compression. Let:

$$\delta_i = \begin{cases} 1 & \text{if } x_i \text{ is in the compression} \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in [1 \ldots n]$$

If we were using a unigram language model, our objective function would maximize the overall sum of the decision variables (i.e., words) multiplied by their unigram probabilities (all probabilities throughout this paper are log-transformed):

$$\max \sum_{i=1}^{n} \delta_i \cdot P(x_i) \tag{1}$$

Thus, if a word is selected, its corresponding δ_i is given a value of 1, and its probability $P(x_i)$ according to the language model will be counted in our total score.

A unigram language model will probably generate many ungrammatical compressions. We therefore use a more context-aware model in our objective function, namely a trigram model. Dynamic programming would be typically used to decode a language model by traversing the sentence in a left-to-right manner. Such an algorithm is efficient and provides all the context required for a conventional language model. However, it can be difficult or impossible to incorporate global constraints into such a model as decisions on word inclusion cannot extend beyond a three word window. By formulating the decoding process for a trigram language model as an integer linear program we are able to take into account constraints that affect the compressed sentence more globally. This process is a much more involved task than in the unigram case where there is no context, instead we must now make decisions based on word sequences rather than isolated words. We first create some additional decision variables:

$$\alpha_i = \begin{cases} 1 & \text{if } x_i \text{ starts the compression} \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in [1 \ldots n]$$

$$\beta_{ij} = \begin{cases} 1 & \text{if sequence } x_i, x_j \text{ ends} \\ & \text{the compression} \qquad \forall i \in [0 \ldots n-1] \\ 0 & \text{otherwise} \qquad \forall j \in [i+1 \ldots n] \end{cases}$$

$$\gamma_{ijk} = \begin{cases} 1 & \text{if sequence } x_i, x_j, x_k \quad \forall i \in [0 \ldots n-2] \\ & \text{is in the compression} \quad \forall j \in [i+1 \ldots n-1] \\ 0 & \text{otherwise} \qquad \forall k \in [j+1 \ldots n] \end{cases}$$

Our objective function is given in Equation (2). This is the sum of all possible trigrams that can occur in all compressions of the source sentence where x_0 represents the 'start' token and x_i is the ith word in sentence $\mathbf{x}$. Equation (3) constrains the decision variables to be binary.

$$\begin{aligned}\max z \quad = \quad & \sum_{i=1}^{n} \alpha_i \cdot P(x_i|\text{start}) \\ & + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} \gamma_{ijk} \cdot P(x_k|x_i, x_j) \\ & + \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} \beta_{ij} \cdot P(\text{end}|x_i, x_j) \qquad (2)\end{aligned}$$

subject to:

$$\delta_i, \alpha_i, \beta_{ij}, \gamma_{ijk} = 0 \text{ or } 1 \qquad (3)$$

The objective function in (2) allows any combination of trigrams to be selected. This means that invalid trigram sequences (e.g., two or more trigrams containing the 'end' token) could appear in the target compression. We avoid this situation by introducing *sequential constraints* (on the decision variables δ_i, γ_{ijk}, α_i, and β_{ij}) that restrict the set of allowable trigram combinations.

Constraint 1 Exactly one word can begin a sentence.

$$\sum_{i=1}^{n} \alpha_i = 1 \qquad (4)$$

Constraint 2 If a word is included in the sentence it must either start the sentence or be preceded by two other words or one other word and the 'start' token x_0.

$$\delta_k - \alpha_k - \sum_{i=0}^{k-2} \sum_{j=1}^{k-1} \gamma_{ijk} = 0 \qquad (5)$$
$$\forall k : k \in [1 \ldots n]$$

Constraint 3 If a word is included in the sentence it must either be preceded by one word and followed by another or it must be preceded by one word and end the sentence.

$$\delta_j - \sum_{i=0}^{j-1} \sum_{k=j+1}^{n} \gamma_{ijk} - \sum_{i=0}^{j-1} \beta_{ij} = 0 \qquad (6)$$
$$\forall j : j \in [1 \ldots n]$$

Constraint 4 If a word is in the sentence it must be followed by two words or followed by one word and then the end of the sentence or it must be preceded by one word and end the sentence.

$$\delta_i - \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} \gamma_{ijk} - \sum_{j=i+1}^{n} \beta_{ij} - \sum_{h=0}^{i-1} \beta_{hi} = 0 \qquad (7)$$
$$\forall i : i \in [1 \ldots n]$$

Constraint 5 Exactly one word pair can end the sentence.

$$\sum_{i=0}^{n-1} \sum_{j=i+1}^{n} \beta_{ij} = 1 \tag{8}$$

The sequential constraints described above ensure that the second order factorization (for trigrams) holds and are different from our compression-specific constraints which are presented in Section 3.5.

Unless normalized by sentence length, a language model will naturally prefer one-word output. This normalization is however non-linear and cannot be incorporated into our ILP formulation. Instead, we impose a constraint on the length of the compressed sentence. Equation (9) below forces the compression to contain at least b tokens.

$$\sum_{i=1}^{n} \delta_i \geq b \tag{9}$$

Alternatively, we could force the compression to be exactly b tokens (by substituting the inequality with an equality in (9)) or to be less than b tokens (by replacing $\geq$ with $\leq$).[4] The constraint in (9) is language model-specific and is not used elsewhere.

3.4.2 Significance Model

The language model just described has no notion of which content words to include in the compression and thus prefers words it has seen before. But words or constituents will be of different relative importance in different documents or even sentences.

Inspired by Hori and Furui (2004), we add to our objective function (see Equation (2)) a significance score designed to highlight important content words. In Hori and Furui's original formulation each word is weighted by a score similar to un-normalized *tf* $*$ *idf*. The significance score is not applied indiscriminately to all words in a sentence but solely to topic-related words, namely nouns and verbs. Our score differs in one respect. It combines document-level with sentence-level significance. So in addition to *tf* $*$ *idf*, each word is weighted by its level of embedding in the syntactic tree.

Intuitively, in a sentence with multiply nested clauses, more deeply embedded clauses tend to carry more semantic content. This is illustrated in Figure 2 which depicts the clause embedding for the sentence "*Mr Field has said he will resign if he is not reselected, a move which could divide the party nationally*". Here, the most important information is conveyed by clauses S_3 (*he will resign*) and S_4 (*if he is not reselected*) which are embedded. Accordingly, we should give more weight to words found in these clauses than in the main clause (S_1 in Figure 2). A simple way to enforce this is to give clauses weight proportional to the level of embedding. Our modified significance score becomes:

$$I(x_i) = \frac{l}{N} \cdot f_i \log \frac{F_a}{F_i} \tag{10}$$

where x_i is a topic word, 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

[4] Compression rate can be also limited to a range by including two inequality constraints.

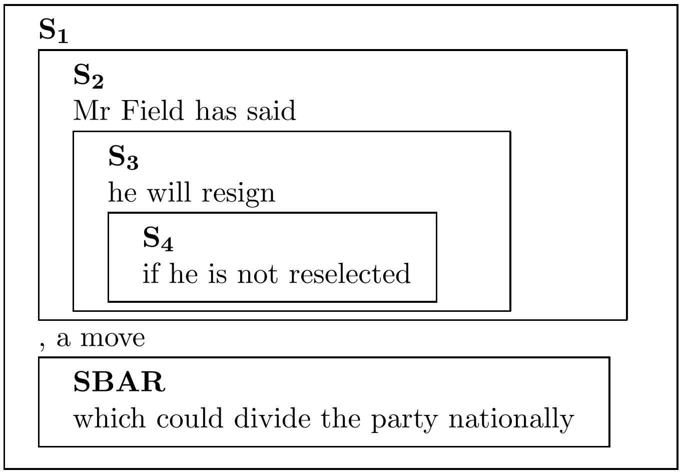

Figure 2: The clause embedding of the sentence "*Mr Field has said he will resign if he is not reselected, a move which could divide the party nationally*"; nested boxes correspond to nested clauses.

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. So, in Figure 2 the term $\frac{l}{N}$ is 1.0 (4/4) for clause S_4, 0.75 (3/4) for clause S_3, and so on. Individual words inherit their weight from their clauses.

The modified objective function with the significance score is given below:

$$\begin{aligned}\max z = & \sum_{i=1}^{n} \delta_i \cdot \lambda I(x_i) + \sum_{i=1}^{n} \alpha_i \cdot P(x_i|\text{start}) \\ & + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} \gamma_{ijk} \cdot P(x_k|x_i, x_j) \\ & + \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} \beta_{ij} \cdot P(\text{end}|x_i, x_j) \end{aligned} \tag{11}$$

We also add a weighting factor (λ) to the objective, in order to counterbalance the importance of the language model and the significance score. The weight is tuned on a small parallel corpus. The sequential constraints from Equations (4)–(8) are again used to ensure that the trigrams are combined in a valid way.

3.4.3 Discriminative Model

As a fully supervised model, we used the discriminative model presented by McDonald (2006). This model uses a large-margin learning framework coupled with a feature set defined on compression bigrams and syntactic structure.

Let $\mathbf{x} = x_1, \ldots, x_n$ denote a source sentence with a target compression $\mathbf{y} = y_1, \ldots, y_m$ where each y_j occurs in $\mathbf{x}$. The function $L(y_i) \in \{1 \ldots n\}$ maps word y_i in the target compression to the index of the word in the source sentence, $\mathbf{x}$. We also include the constraint that $L(y_i) < L(y_{i+1})$ which forces each word in $\mathbf{x}$ to occur at most once in the compression $\mathbf{y}$. Let the score of a compression $\mathbf{y}$ for a sentence $\mathbf{x}$ be:

$$s(\mathbf{x}, \mathbf{y}) \tag{12}$$

This score is factored using a first-order Markov assumption on the words in the target compression to give:

$$s(\mathbf{x}, \mathbf{y}) = \sum_{j=2}^{|\mathbf{y}|} s(\mathbf{x}, L(y_{j-1}), L(y_j)) \tag{13}$$

The score function is defined to be the dot product between a high dimensional feature representation and a corresponding weight vector:

$$s(\mathbf{x}, \mathbf{y}) = \sum_{j=2}^{|\mathbf{y}|} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, L(y_{j-1}), L(y_j)) \tag{14}$$

Decoding in this model amounts to finding the combination of bigrams that maximizes the scoring function in (14). McDonald (2006) uses a dynamic programming approach where the maximum score is found in a left-to-right manner. The algorithm is an extension of Viterbi for the case in which scores factor over dynamic sub-strings (Sarawagi & Cohen, 2004; McDonald, Crammer, & Pereira, 2005a). This allows back-pointers to be used to reconstruct the highest scoring compression as well as the k-best compressions.

Again this is similar to the trigram language model decoding process (see Section 3.4.1), except that here a bigram model is used. Consequently, the ILP formulation is slightly simpler than that of the trigram language model. Let:

$$\delta_i = \begin{cases} 1 & \text{if } x_i \text{ is in the compression} \\ 0 & \text{otherwise} \end{cases} \quad (1 \leq i \leq n)$$

We then introduce some more decision variables:

$$\alpha_i = \begin{cases} 1 & \text{if } x_i \text{ starts the compression} \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in [1 \ldots n]$$

$$\beta_i = \begin{cases} 1 & \text{if word } x_i \text{ ends the compression} \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in [1 \ldots n]$$

$$\gamma_{ij} = \begin{cases} 1 & \text{if sequence } x_i, x_j \text{ is in the compression} \\ 0 & \text{otherwise} \end{cases} \quad \begin{array}{l} \forall i \in [1 \ldots n-1] \\ \forall j \in [i+1 \ldots n] \end{array}$$

The discriminative model can be now expressed as:

$$\begin{aligned} \max z \quad = \quad & \sum_{i=1}^{n} \alpha_i \cdot s(\mathbf{x}, 0, i) \\ & + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \gamma_{ij} \cdot s(\mathbf{x}, i, j) \\ & + \sum_{i=1}^{n} \beta_i \cdot s(\mathbf{x}, i, n+1) \end{aligned} \tag{15}$$

Constraint 1 Exactly one word can begin a sentence.

$$\sum_{i=1}^{n} \alpha_i = 1 \tag{16}$$

Constraint 2 If a word is included in the sentence it must either start the compression or follow another word.

$$\delta_j - \alpha_j - \sum_{i=1}^{j} \gamma_{ij} = 0 \tag{17}$$
$$\forall j : j \in [1 \ldots n]$$

Constraint 3 If a word is included in the sentence it must be either followed by another word or end the sentence.

$$\delta_i - \sum_{j=i+1}^{n} \gamma_{ij} - \beta_i = 0 \tag{18}$$
$$\forall i : i \in [1 \ldots n]$$

Constraint 4 Exactly one word can end a sentence.

$$\sum_{i=1}^{n} \beta_i = 1 \tag{19}$$

Again, the sequential constraints in Equations (16)–(19) are necessary to ensure that the resulting combination of bigrams are valid.

The current formulation provides a single optimal compression given the model. However, McDonald's (2006) dynamic programming algorithm is capable of returning the k-best compressions; this is useful for their learning algorithm described later. In order to produce k-best compressions, we must rerun the ILP with extra constraints which forbid previous solutions. In other words, we first formulate the ILP as above, solve it, add its solution to the k-best list, and then create a set of constraints that forbid the configuration of δ_i decision variables which form the current solution. The procedure is repeated until k compressions are found.

The computation of the compression score crucially relies on the dot product between a high dimensional feature representation and a corresponding weight vector (see Equation (14)). McDonald (2006) employs a rich feature set defined over adjacent words and individual parts-of-speech, dropped words and phrases from the source sentence, and dependency structures (also of the source sentence). These features are designed to mimic the information presented in the previous noisy-channel and decision-tree models of Knight and Marcu (2002). Features over adjacent words are used as a proxy to the language model of the noisy channel. Unlike other models, which treat the parses as gold standard, McDonald uses the dependency information as another form of evidence. Faced with parses that are noisy the learning algorithm can reduce the weighting given to those features if they prove poor discriminators on the training data. Thus, the model should be much more robust and portable across different domains and training corpora.

The weight vector, $\mathbf{w}$ is learned using the Margin Infused Relaxed Algorithm (MIRA, Crammer & Singer, 2003) a discriminative large-margin online learning technique (McDonald, Crammer, & Pereira, 2005b). This algorithm learns by compressing each sentence and comparing the result with the gold standard. The weights are updated so that the score of the correct compression (the gold standard) is greater than the score of all other compressions by a margin proportional to their loss. The loss function is the number of words falsely retained or dropped in the incorrect compression relative to the gold standard. A source sentence will have exponentially many compressions and thus exponentially many margin constraints. To render learning computationally tractable, McDonald et al. (2005b) create constraints only on the k compressions that currently have the highest score, $\text{best}_k(\mathbf{x}; \mathbf{w})$.

3.5 Constraints

We are now ready to describe our compression-specific constraints. The models presented in the previous sections contain only sequential constraints and are thus equivalent to their original formulation. Our constraints are linguistically and semantically motivated in a similar fashion to the grammar checking component of Jing (2000). However, they do not rely on any additional knowledge sources (such as a grammar lexicon or WordNet) beyond the parse and grammatical relations of the source sentence. We obtain these from RASP (Briscoe & Carroll, 2002), a domain-independent, robust parsing system for English. However, any other parser with broadly similar output (e.g., Lin, 2001) could also serve our purposes. Our constraints revolve around modification, argument structure, and discourse related factors.

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

$$\delta_i - \delta_j \geq 0 \tag{20}$$
$$\forall i, j : x_j \in x_i\text{'s } \texttt{ncmods}$$

$$\delta_i - \delta_j \geq 0 \tag{21}$$
$$\forall i, j : x_j \in x_i\text{'s } \texttt{detmods}$$

Equation (20) guarantees that if we include a non-clausal modifier[5] (`ncmod`) in the compression (such as an adjective or a noun) then the head of the modifier must also be included; this is repeated for determiners (`detmod`) in (21). In Table 2 we illustrate how these constraints disallow the deletion of certain words (starred sentences denote compressions that would not be possible given our constraints). For example, if the modifier word *Pasok* from sentence (1a) is in the compression, then its head *Party* will also included (see (1b)).

We also want to ensure that the meaning of the source sentence is preserved in the compression, particularly in the face of negation. Equation (22) implements this by forcing *not* in the compression when the head is included (see sentence (2b) in Table 2). A similar constraint is added for possessive modifiers (e.g., *his*, *our*), including genitives (e.g., *John's*

[5] Clausal modifiers (`cmod`) are adjuncts modifying entire clauses. In the example "*he ate the cake because he was hungry*", the *because*-clause is a modifier of the sentence "*he ate the cake*".

1a. He became a power player in Greek Politics in 1974, when he founded the socialist Pasok Party.

1b. *He became a power player in Greek Politics in 1974, when he founded **the Pasok**.

2a. We took these troubled youth who don't have fathers, and brought them into a room to Dads who don't have their children.

2b. *We took these troubled youth who **do have** fathers, and brought them into a room to Dads who **do have** their children.

2c. *We took these troubled youth who don't have fathers, and brought them into a room to Dads who don't **have children**.

3a. The chain stretched from Uganda to Grenada and Nicaragua, since the 1970s.

3b. ***Stretched from Uganda** to Grenada and Nicaragua, since the 1970s.

3c. ***The chain from Uganda** to Grenada and Nicaragua, since the 1970s.

3d. *The chain **stretched Uganda** to Grenada and Nicaragua, since the 1970s.

3e. *The chain **stretched from** to Grenada and Nicaragua, since the 1970s.

3f. *The chain stretched from **Uganda to Grenada Nicaragua**, since the 1970s.

Table 2: Examples of compressions disallowed by our set of constraints.

gift), as shown in Equation (23). An example of the possessive constraint is given in sentence (2c) in Table 2.

$$\delta_i - \delta_j = 0 \tag{22}$$
$$\forall i, j : x_j \in x_i\text{'s } \texttt{ncmods} \wedge x_j = \text{not}$$
$$\delta_i - \delta_j = 0 \tag{23}$$
$$\forall i, j : x_j \in x_i\text{'s possessive } \texttt{mods}$$

Argument Structure Constraints We also define a few intuitive constraints that take the overall sentence structure into account. The first constraint (Equation (24)) ensures that if a verb is present in the compression then so are its arguments, and if any of the arguments are included in the compression then the verb must also be included. We thus force the program to make the same decision on the verb, its subject, and object (see sentence (3b) in Table 2).

$$\delta_i - \delta_j = 0 \tag{24}$$
$$\forall i, j : x_j \in \text{subject/object of verb } x_i$$

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

$$\sum_{i: x_i \in verbs} \delta_i \geq 1 \tag{25}$$

The constraint entails that it is not possible to drop the main verb *stretched* from sentence (3a) (see also sentence (3c) in Table 2).

Other sentential constraints include Equations (26) and (27) which apply to prepositional phrases and subordinate clauses. These constraints force the introducing term (i.e., the preposition, or subordinator) to be included in the compression if any word from within the syntactic constituent is also included. By subordinator we mean *wh*-words (e.g., *who, which, how, where*), the word *that*, and subordinating conjunctions (e.g., *after, although, because*). The reverse is also true, i.e., if the introducing term is included, at least one other word from the syntactic constituent should also be included.

$$\begin{aligned} & \delta_i - \delta_j \geq 0 \qquad (26) \\ \forall i, j : x_j \in\ & \texttt{PP/SUB} \\ \wedge x_i \text{ starts } & \texttt{PP/SUB} \\ \sum_{i:x_i \in \texttt{PP/SUB}} & \delta_i - \delta_j \geq 0 \qquad (27) \\ \forall j : x_j \text{ starts } & \texttt{PP/SUB} \end{aligned}$$

As an example consider sentence (3d) from Table 2. Here, we cannot drop the preposition *from* if *Uganda* is in the compression. Conversely, we must include *from* if *Uganda* is in the compression (see sentence (3e)).

We also wish to handle coordination. 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:

$$\begin{aligned} (1 - \delta_i) + \delta_j &\geq 1 \qquad (28) \\ (1 - \delta_i) + \delta_k &\geq 1 \qquad (29) \\ \delta_i + (1 - \delta_j) + (1 - \delta_k) &\geq 1 \qquad (30) \\ \forall i, j, k : x_j \wedge x_k & \text{ conjoined by } x_i \end{aligned}$$

Consider sentence (3f) from Table 2. If both *Uganda* and *Nicaragua* are present in the compression, then we must include the conjunction *and*.

Finally, Equation (31) disallows anything within brackets in the source sentence from being included in the compression. This is a somewhat superficial attempt at excluding parenthetical and potentially unimportant material from the compression.

$$\begin{aligned} \delta_i &= 0 \qquad (31) \\ \forall i : x_i \in\ & \text{bracketed words (inc parentheses)} \end{aligned}$$

Discourse Constraints Our discourse constraint concerns personal pronouns. Specifically, Equation (32) forces personal pronouns to be included in the compression. The constraint is admittedly more important for generating coherent documents (as opposed to individual sentences). It nevertheless has some impact on sentence-level compressions, in particular when verbal arguments are missed by the parser. When these are pronominal, constraint (32) will result in more grammatical output since some of the argument structure of the source sentence will be preserved in the compression.

$$\begin{aligned} \delta_i &= 1 \qquad (32) \\ \forall i : x_i \in\ & \text{personal pronouns} \end{aligned}$$

We should note that some of the constraints described above would be captured by models that learn synchronous deletion rules from a corpus. For example, the noisy-channel model of Knight and Marcu (2002) learns not to drop the head when the latter is modified by an adjective or a noun, since the transformations *DT NN* → *DT* or *AJD NN* → *ADJ* are almost never seen in the data. Similarly, the coordination constraint (Equations (28)–(30)) would be enforced using Turner and Charniak's (2005) special rules — they enhance their parallel grammar with rules modeling more structurally complicated deletions than those attested in their corpus. In designing our constraints we aimed at capturing appropriate deletions for many possible models, including those that do not rely on a training corpus or do not have an explicit notion of a parallel grammar (e.g., McDonald, 2006). The modification constraints would presumably be redundant for the noisy-channel model, which could otherwise benefit from more specialized constraints, e.g., targeting sparse rules or noisy parse trees, however we leave this to future work.

Another feature of the modeling framework presented here is that deletions (or non-deletions) are treated as unconditional decisions. For example, we require not to drop the noun in adjective-noun sequences if the adjective is not deleted as well. We also require to always include a verb in the compression if the source sentence has one. These hardwired decisions could in some cases prevent valid compressions from being considered. For instance, it is not possible to compress the sentence "*this is not appropriate behavior*" to "*this is not appropriate*" or"*Bob loves Mary and John loves Susan*" to "*Bob loves Mary and John Susan*". Admittedly we lose some expressive power, yet we ensure that the compressions will be broadly grammatically, even for unsupervised or semi-supervised models. Furthermore, in practice we find that our models consistently outperform non-constraint-based alternatives, without extensive constraint engineering.

3.6 Solving the ILP

As we mentioned earlier (Section 3.1), solving ILPs is NP-hard. In cases where the coefficient matrix is unimodular, it can be shown that the optimal solution to the linear program is integral. Although the coefficient matrix in our problems is not unimodular, we obtained integral solutions for all sentences we experimented with (approximately 3,000, see Section 4.1 for details). We conjecture that this is due to the fact that all of our variables have 0, $+1$ or -1 coefficients in the constraints and therefore our constraint matrix shares many properties of a unimodular matrix. We generate and solve an ILP for every sentence we wish to compress. Solve times are less than a second per sentence (including input-output overheads) for all models presented here.

4. Experimental Set-up

Our evaluation experiments were motivated by three questions: (1) Do the constraint-based compression models deliver performance gains over non-constraint-based ones? We expect better compressions for the model variants which incorporate compression-specific constraints. (2) Are there differences among constraint-based models? Here, we would like to investigate how much modeling power is gained by the addition of the constraints. For example, it may be the case that a state-of-the-art model like McDonald's (2006) does not benefit much from the addition of constraints. And that their effect is much bigger for less sophisticated models. (3) How do the models reported in this paper port across domains? In particular, we are interested in assessing whether the models and proposed constraints are general and robust enough to produce good compressions for both written and spoken texts.

We next describe the data sets on which our models were trained and tested (Section 4.1), explain how model parameters were estimated (Section 4.2) and present our evaluation setup (Section 4.3). We discuss our results in Section 5.

4.1 Corpora

Our intent was to assess the performance of the models just described on written and spoken text. The appeal of written text is understandable since most summarization work today focuses on this domain. Speech data not only provides a natural test-bed for compression applications (e.g., subtitle generation) but also poses additional challenges. Spoken utterances can be ungrammatical, incomplete, and often contain artefacts such as false starts, interjections, hesitations, and disfluencies. Rather than focusing on spontaneous speech which is abundant in these artefacts, we conduct our study on the less ambitious domain of broadcast news transcripts. This lies in-between the extremes of written text and spontaneous speech as it has been scripted beforehand and is usually read off on autocue.

Previous work on sentence compression has almost exclusively used the Ziff-Davis corpus for training and testing purposes. This corpus originates from a collection of news articles on computer products. It was created automatically by matching sentences that occur in an article with sentences that occur in an abstract (Knight & Marcu, 2002). The abstract sentences had to contain a subset of the source sentence's words and the word order had to remain the same. In earlier work (Clarke & Lapata, 2006) we have argued that the Ziff-Davis corpus is not ideal for studying compression for several reasons. First, we showed that human-authored compressions differ substantially from the Ziff-Davis which tends to be more aggressively compressed. Second, humans are more likely to drop individual words than lengthy constituents. Third, the test portion of the Ziff-Davis contains solely 32 sentences. This is an extremely small data set to reveal any statistically significant differences among systems. In fact, previous studies relied almost exclusively on human judgments for assessing the well-formedness of the compressed output, and significance tests are reported for by-subjects analyses only.

We thus focused in the present study on manually created corpora. Specifically, we asked annotators to perform sentence compression by removing tokens on a sentence-by-sentence basis. Annotators were free to remove any words they deemed superfluous provided their deletions: (a) preserved the most important information in the source sentence, and (b) ensured the compressed sentence remained grammatical. If they wished, they could leave a sentence uncompressed by marking it as inappropriate for compression. They were not allowed to delete whole sentences even if they believed they contained no information content with respect to the story as this would blur the task with abstracting. Following these guidelines, our annotators produced compressions of 82 newspaper articles (1,433 sentences) from the British National Corpus (BNC) and the American News Text corpus (henceforth written corpus) and 50 stories (1,370 sentences) from the HUB-4 1996 English Broadcast News corpus (henceforth spoken corpus). The written corpus contains articles from The LA

Times, Washington Post, Independent, The Guardian and Daily Telegraph. The spoken corpus contains broadcast news from a variety of networks (CNN, ABC, CSPAN and NPR) which have been manually transcribed and segmented at the story and sentence level. Both corpora have been split into training, development and testing sets[6] randomly on article boundaries (with each set containing full stories) and are publicly available from `http://homepages.inf.ed.ac.uk/s0460084/data/`.

4.2 Parameter Estimation

In this work we present three compression models ranging from unsupervised to semi-supervised, and fully supervised. The unsupervised model simply relies on a trigram language model for driving compression (see Section 3.4.1). This was estimated from 25 million tokens of the North American corpus using the CMU-Cambridge Language Modeling Toolkit (Clarkson & Rosenfeld, 1997) with a vocabulary size of 50,000 tokens and Good-Turing discounting. To discourage one-word output we force the ILP to generate compressions whose length is no less than 40% of the source sentence (see the constraint in (9)). The semi-supervised model is the weighted combination of a word-based significance score with a language model (see Section 3.4.2). The significance score was calculated using 25 million tokens from the American News Text corpus. We optimized its weight (see Equation (11)) on a small subset of the training data (three documents in each case) using Powell's method (Press, Teukolsky, Vetterling, & Flannery, 1992) and a loss function based on the F-score of the grammatical relations found in the gold standard compression and the system's best compression (see Section 4.3 for details). The optimal weight was approximately 1.8 for the written corpus and 2.2 for the spoken corpus.

McDonald's (2006) supervised model was trained on the written and spoken training sets. Our implementation used the same feature sets as McDonald, the only difference being that our phrase structure and dependency features were extracted from the output of Roark's (2001) parser. McDonald uses Charniak's (2000) parser which performs comparably. The model was learnt using k-best compressions. On the development data, we found that $k = 10$ provided the best performance.

4.3 Evaluation

Previous studies have relied almost exclusively on human judgments for assessing the well-formedness of automatically derived compressions. These are typically rated by naive subjects on two dimensions, grammaticality and importance (Knight & Marcu, 2002). Although automatic evaluation measures have been proposed (Riezler et al., 2003; Bangalore, Rambow, & Whittaker, 2000) their use is less widespread, we suspect due to the small size of the test portion of the Ziff-Davis corpus which is commonly used in compression work.

We evaluate the output of our models in two ways. First, we present results using an automatic evaluation measure put forward by Riezler et al. (2003). They compare the grammatical relations found in the system compressions against those found in a gold standard. This allows us to measure the semantic aspects of summarization quality in terms of grammatical-functional information and can be quantified using F-score. Furthermore,

[6] The splits are 908/63/462 sentences for the written corpus and 882/78/410 sentences for the spoken corpus.

in Clarke and Lapata (2006) we show that relations-based F-score correlates reliably with human judgments on compression output. Since our test corpora are larger than Ziff-Davis (by more than a factor of ten), differences among systems can be highlighted using significance testing.

Our implementation of the F-score measure used the grammatical relations annotations provided by RASP (Briscoe & Carroll, 2002). This parser is particularly appropriate for the compression task since it provides parses for both full sentences and sentence fragments and is generally robust enough to analyze semi-grammatical sentences. We calculated F-score over all the relations provided by RASP (e.g., subject, direct/indirect object, modifier; 15 in total).

In line with previous work we also evaluate our models by eliciting human judgments. Following the work of Knight and Marcu (2002), we conducted two separate experiments. In the first experiment participants were presented with a source sentence and its target compression and asked to rate how well the compression preserved the most important information from the source sentence. In the second experiment, they were asked to rate the grammaticality of the compressed outputs. In both cases they used a five point rating scale where a high number indicates better performance. We randomly selected 21 sentences from the test portion of each corpus. These sentences were compressed automatically by the three models presented in this paper with and without constraints. We also included gold standard compressions. Our materials thus consisted of 294 ($21 \times 2 \times 7$) source-target sentences. A Latin square design ensured that subjects did not see two different compressions of the same sentence. We collected ratings from 42 unpaid volunteers, all self reported native English speakers. Both studies were conducted over the Internet using a custom build web interface. Examples of our experimental items are given in Table 3.

5. Results

Let us first discuss our results when compression output is evaluated in terms of F-score. Tables 4 and 5 illustrate the performance of our models on the written and spoken corpora, respectively. We also present the compression rate[7] for each system. In all cases the constraint-based models (+Constr) yield better F-scores than the non-constrained ones. The difference is starker for the semi-supervised model (Sig). The constraints bring an improvement of 17.2% on the written corpus and 18.3% on the spoken corpus. We further examined whether performance differences among models are statistically significant, using the Wilcoxon test. On the written corpus all constraint models significantly outperform the models without constraints. The same tendency is observed on the spoken corpus except for the model of McDonald (2006) which performs comparably with and without constraints.

We also wanted to establish which is the best constraint model. On both corpora we find that the language model performs worst, whereas the significance model and McDonald perform comparably (i.e., the F-score differences are not statistically significant). To get a feeling for the difficulty of the task, we calculated how much our annotators agreed in their compression output. The inter-annotator agreement (F-score) on the written corpus was 65.8% and on the spoken corpus 73.4%. The agreement is higher on spoken texts since they consists of many short utterances (e.g., *Okay*, *That's it for now*, *Good night*) that can

[7] The term refers to the percentage of words *retained* from the source sentence in the compression.

Source	The aim is to give councils some control over the future growth of second homes.
Gold	The aim is to give councils control over the growth of homes.
LM	The aim is to the future.
LM+Constr	The aim is to give councils control.
Sig	The aim is to give councils control over the future growth of homes.
Sig+Constr	The aim is to give councils control over the future growth of homes.
McD	The aim is to give councils.
McD+Constr	The aim is to give councils some control over the growth of homes.
Source	The Clinton administration recently unveiled a new means to encourage brownfields redevelopment in the form of a tax incentive proposal.
Gold	The Clinton administration unveiled a new means to encourage brownfields redevelopment in a tax incentive proposal.
LM	The Clinton administration in the form of tax.
LM+Constr	The Clinton administration unveiled a means to encourage redevelopment in the form.
Sig	The Clinton administration unveiled a encourage brownfields redevelopment form tax proposal.
Sig+Constr	The Clinton administration unveiled a means to encourage brownfields redevelopment in the form of tax proposal.
McD	The Clinton unveiled a means to encourage brownfields redevelopment in a tax incentive proposal.
McD+Constr	The Clinton administration unveiled a means to encourage brownfields redevelopment in the form of a incentive proposal.

Table 3: Example compressions produced by our systems (Source: source sentence, Gold: gold-standard compression, LM: language model compression, LM+Constr: language model compression with constraints, Sig: significance model, Sig+Constr: significance model with constraints, McD: McDonald's (2006) compression model, McD+Constr: McDonald's (2006) compression model with constraints).

be compressed only very little or not all. Note that there is a marked difference between the automatic and human compressions. Our best performing systems are inferior to human output by more than 20 F-score percentage points.

Differences between the automatic systems and the human output are also observed with respect to the compression rate. As can be seen the language model compresses most aggressively, whereas the significance model and McDonald tend to be more conservative and closer to the gold standard. Interestingly, the constraints do not necessarily increase the compression rate. The latter increases for the significance model but decreases for the language model and remains relatively constant for McDonald. It is straightforward to impose the same compression rate for all constraint-based models (e.g., by forcing the model to retain b tokens $\sum_{i=1}^{n} \delta_i = b$). However, we refrained from doing this since we wanted our

Models	CompR	F-score
LM	46.2	18.4
Sig	60.6	23.3
McD	60.1	36.0
LM+Constr	41.2	28.2*
Sig+Constr	72.0	40.5*†
McD+Constr	63.7	40.8*†
Gold	70.3	—

Table 4: Results on the written corpus; compression rate (CompR) and grammatical relation F-score (F-score); *: +Constr model is significantly different from model without constraints; †: significantly different from LM+Constr.

Models	CompR	F-score
LM	52.0	25.4
Sig	60.9	30.4
McD	68.6	47.6
LM+Constr	49.5	34.8*
Sig+Constr	78.4	48.7*†
McD+Constr	68.5	50.1†
Gold	76.1	—

Table 5: Results on the spoken corpus; compression rate (CompR) and grammatical relation F-score (F-score); *: +Constr model is significantly different from without constraints; †: significantly different from LM+Constr.

models to regulate the compression rate for each sentence individually according to its specific information content and structure.

We next consider the results of our human study which assesses in more detail the quality of the generated compressions on two dimensions, namely grammaticality and information content. F-score conflates these two dimensions and therefore in theory could unduly reward a system that produces perfectly grammatical output without any information loss. Tables 6 and 7 show the mean ratings[8] for each system (and the gold standard) on the written and spoken corpora, respectively. We first performed an Analysis of Variance (Anova) to examine the effect of different system compressions. The Anova revealed a reliable effect on both grammaticality and importance for each corpus (the effect was significant by both subjects and items ($p < 0.01$)).

We next examine the impact of the constraints (+Constr in the tables). In most cases we observe an increase in ratings for both grammaticality and importance when a model is supplemented constraints. Post-hoc Tukey tests reveal that the grammaticality and importance ratings of the language model and significance model significantly improve with

[8] All statistical tests reported subsequently were done using the mean ratings.

Models	Grammar	Importance
LM	$2.25^{\dagger\$}$	$1.82^{\dagger\$}$
Sig	$2.26^{\dagger\$}$	$2.99^{\dagger\$}$
McD	$3.05^{\dagger}$	$2.84^{\dagger}$
LM+Constr	$3.47^{*\dagger}$	$2.37^{*\dagger\$}$
Sig+Constr	3.76^{*}	3.53^{*}
McD+Constr	$3.50^{\dagger}$	$3.17^{\dagger}$
Gold	4.25	3.98

Table 6: Results on the written text corpus; average grammaticality score (Grammar) and average importance score (Importance) for human judgments; *: +Constr model is significantly different from model without constraints; †: significantly different from gold standard; $^\$$: significantly different from McD+Constr.

Models	Grammar	Importance
LM	$2.20^{\dagger\$}$	$1.56^{\dagger}$
Sig	$2.29^{\dagger\$}$	$2.64^{\dagger}$
McD	$3.33^{\dagger}$	$3.32^{\dagger}$
LM+Constr	$3.18^{*\dagger}$	$2.49^{*\dagger\$}$
Sig+Constr	$3.80^{*\dagger}$	$3.69^{*\dagger}$
McD+Constr	$3.60^{\dagger}$	$3.31^{\dagger}$
Gold	4.45	4.25

Table 7: Results on the spoken text corpus; average grammaticality score (Grammar) and average importance score (Importance) for human judgments; *: +Constr model is significantly different from model without constraints; †: significantly different from gold standard; $^\$$: significantly different from McD+Constr.

the constraints ($\alpha < 0.01$). In contrast, McDonald's system sees a numerical improvement with the additional constraints, but this difference is not statistically significant. These tendencies are observed on the spoken and written corpus.

Upon closer inspection, we can see that the constraints influence considerably the grammaticality of the unsupervised and semi-supervised systems. Tukey tests reveal that LM+Constr and Sig+Constr are as grammatical as McD+Constr. In terms of importance, Sig+Constr and McD+Constr are significantly better than LM+Constr ($\alpha < 0.01$). This is not surprising given that LM+Constr is a very simple model without a mechanism for highlighting important words in a sentence. Interestingly, Sig+Constr performs as well as McD+Constr in retaining the most important words, despite the fact that it requires minimal supervision. Although constraint-based models overall perform better than models without constraints, they receive lower ratings (for grammaticality and importance) in comparison to the gold standard. And the differences are significant in most cases.

In summary, we observe that the constraints boost performance. This is more pronounced for compression models that are either unsupervised or use small amounts of parallel data. For example, a simple model like Sig yields performance comparable to McDonald (2006) when constraints are taken into account. This is an encouraging result suggesting that ILP can be used to create good compression models with relatively little effort (i.e., without extensive feature engineering or elaborate knowledge sources). Performance gains are also obtained for competitive models like McDonald's that are fully supervised. But these gains are smaller, presumably because the initial model contains a rich feature representation consisting of syntactic information and generally does a good job at producing grammatical output. Finally, our improvements are consistent across corpora and evaluation paradigms.

6. Conclusions

In this paper we have presented a novel method for automatic sentence compression. A key aspect of our approach is the use of integer linear programming for inferring globally optimal compressions in the presence of linguistically motivated constraints. We have shown how previous formulations of sentence compression can be recast as ILPs and extended these models with local and global constraints ensuring that the compressed output is structurally and semantic well-formed. Contrary to previous work that has employed ILP solely for decoding, our models integrate learning with inference in a unified framework.

Our experiments have demonstrated the advantages of the approach. Constraint-based models consistently bring performance gains over models without constraints. These improvements are more impressive for models that require little or no supervision. A case in point here is the significance model discussed above. The no-constraints incarnation of this model performs poorly and considerably worse than McDonald's (2006) state-of-the-art model. The addition of constraints improves the output of this model so that its performance is indistinguishable from McDonald. Note that the significance model requires a small amount of training data (50 parallel sentences), whereas McDonald is trained on hundreds of sentences. It also presupposes little feature engineering, whereas McDonald utilizes thousands of features. Some effort is associated with framing the constraints, however these are created once and are applied across models and corpora. We have also observed small performance gains for McDonald's system when the latter is supplemented with constraints. Larger improvements are possible with more sophisticated constraints, however our intent was to devise a set of general constraints that are not tuned to the mistakes of any specific system in particular.

Future improvements are many and varied. An obvious extension concerns our constraint set. Currently our constraints are mostly syntactic and consider each sentence in isolation. By incorporating discourse constraints we could highlight words that are important at the document-level. Presumably words topical in a document should be retained in the compression. Other constraints could manipulate the compression rate. For example, we could encourage a higher compression rate for longer sentences. Another interesting direction includes the development of better objective functions for the compression task. The objective functions presented so far rely on first or second-order Markov assumptions. Alternative objectives could take into account the structural similarity between the source sentence and its target compression; or whether they share the same content which could be operationalized in terms of entropy.

Beyond the task and systems presented in this paper, we believe the approach holds promise for other generation applications using decoding algorithms for searching the space of possible outcomes. Examples include sentence-level paraphrasing, headline generation, and summarization.

Acknowledgments

We are grateful to our annotators Vasilis Karaiskos, Beata Kouchnir, and Sarah Luger. Thanks to Jean Carletta, Frank Keller, Steve Renals, and Sebastian Riedel for helpful comments and suggestions and to the anonymous referees whose feedback helped to substantially improve the present paper. Lapata acknowledges the support of EPSRC (grant GR/T04540/01). A preliminary version of this work was published in the proceedings of ACL 2006.

References

Aho, A. V., & Ullman, J. D. (1969). Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, *3*, 37–56.

Bangalore, S., Rambow, O., & Whittaker, S. (2000). Evaluation metrics for generation. In *Proceedings of the first International Conference on Natural Language Generation*, pp. 1–8, Mitzpe Ramon, Israel.

Barzilay, R., & Lapata, M. (2006). Aggregation via set partitioning for natural language generation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 359–366, New York, NY, USA.

Bramsen, P., Deshpande, P., Lee, Y. K., & Barzilay, R. (2006). Inducing temporal graphs. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pp. 189–198, Sydney, Australia.

Briscoe, E. J., & Carroll, J. (2002). Robust accurate statistical annotation of general text. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pp. 1499–1504, Las Palmas, Gran Canaria.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the 1st North American Annual Meeting of the Association for Computational Linguistics*, pp. 132–139, Seattle, WA, USA.

Clarke, J., & Lapata, M. (2006). Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 377–384, Sydney, Australia.

Clarkson, P., & Rosenfeld, R. (1997). Statistical language modeling using the CMU–Cambridge toolkit. In *Proceedings of Eurospeech'97*, pp. 2707–2710, Rhodes, Greece.

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1992). *Intoduction to Algorithms*. The MIT Press.

Corston-Oliver, S. (2001). Text Compaction for Display on Very Small Screens. In *Proceedings of the Workshop on Automatic Summarization at the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 89–98, Pittsburgh, PA, USA.

Crammer, K., & Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, *3*, 951–991.

Dantzig, G. B. (1963). *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, USA.

Denis, P., & Baldridge, J. (2007). Joint determination of anaphoricity and coreference resolution using integer programming. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pp. 236–243, Rochester, NY.

Dras, M. (1999). *Tree Adjoining Grammar and the Reluctant Paraphrasing of Text*. Ph.D. thesis, Macquarie University.

Galley, M., & McKeown, K. (2007). Lexicalized markov grammars for sentence compression. In *In Proceedings of the North American Chapter of the Association for Computational Linguistics*, pp. 180–187, Rochester, NY, USA.

Gomory, R. E. (1960). Solving linear programming problems in integers. In Bellman, R., & Hall, M. (Eds.), *Combinatorial analysis, Proceedings of Symposia in Applied Mathematics*, Vol. 10, Providence, RI, USA.

Grefenstette, G. (1998). Producing Intelligent Telegraphic Text Reduction to Provide an Audio Scanning Service for the Blind. In Hovy, E., & Radev, D. R. (Eds.), *Proceedings of the AAAI Symposium on Intelligent Text Summarization*, pp. 111–117, Stanford, CA, USA.

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

Jing, H. (2000). Sentence reduction for automatic text summarization. In *Proceedings of the 6th Applied Natural Language Processing Conference*, pp. 310–315, Seattle,WA, USA.

Knight, K., & Marcu, D. (2002). Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artificial Intelligence*, *139*(1), 91–107.

Land, A. H., & Doig, A. G. (1960). An automatic method for solving discrete programming problems. *Econometrica*, *28*, 497–520.

Lin, C.-Y. (2003). Improving summarization performance by sentence compression — a pilot study. In *Proceedings of the 6th International Workshop on Information Retrieval with Asian Languages*, pp. 1–8, Sapporo, Japan.

Lin, D. (2001). LaTaT: Language and text analysis tools. In *Proceedings of the first Human Language Technology Conference*, pp. 222–227, San Francisco, CA, USA.

Marciniak, T., & Strube, M. (2005). Beyond the pipeline: Discrete optimization in NLP. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pp. 136–143, Ann Arbor, MI, USA.

McDonald, R. (2006). Discriminative sentence compression with soft syntactic constraints. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy.

McDonald, R., Crammer, K., & Pereira, F. (2005a). Flexible text segmentation with structured multilabel classification. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pp. 987–994, Vancouver, BC, Canada.

McDonald, R., Crammer, K., & Pereira, F. (2005b). Online large-margin training of dependency parsers. In *43rd Annual Meeting of the Association for Computational Linguistics*, pp. 91–98, Ann Arbor, MI, USA.

Nemhauser, G. L., & Wolsey, L. A. (1988). *Integer and Combinatorial Optimization.* Wiley-Interscience series in discrete mathematicals and opitmization. Wiley, New York, NY, USA.

Nguyen, M. L., Shimazu, A., Horiguchi, S., Ho, T. B., & Fukushi, M. (2004). Probabilistic sentence reduction using support vector machines. In *Proceedings of the 20th international conference on Computational Linguistics*, pp. 743–749, Geneva, Switzerland.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, New York, NY, USA.

Punyakanok, V., Roth, D., Yih, W., & Zimak, D. (2004). Semantic role labeling via integer linear programming inference. In *Proceedings of the International Conference on Computational Linguistics*, pp. 1346–1352, Geneva, Switzerland.

Riedel, S., & Clarke, J. (2006). Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pp. 129–137, Sydney, Australia.

Riezler, S., King, T. H., Crouch, R., & Zaenen, A. (2003). Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Human Language Technology Conference and the 3rd Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 118–125, Edmonton, Canada.

Roark, B. (2001). Probabilistic top-down parsing and language modeling. *Computational Linguistics*, *27*(2), 249–276.

Roth, D. (1998). Learning to resolve natural language ambiguities: A unified approach. In *In Proceedings of the 15th of the American Association for Artificial Intelligence*, pp. 806–813, Madison, WI, USA.

Roth, D., & Yih, W. (2004). A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Annual Conference on Computational Natural Language Learning*, pp. 1–8, Boston, MA, USA.

Roth, D., & Yih, W. (2005). Integer linear programming inference for conditional random fields. In *Proceedings of the International Conference on Machine Learning*, pp. 737–744, Bonn.

Sarawagi, S., & Cohen, W. W. (2004). Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, Vancouver, BC, Canada.

Shieber, S., & Schabes, Y. (1990). Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, pp. 253–258, Helsinki, Finland.

Turner, J., & Charniak, E. (2005). Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 290–297, Ann Arbor, MI, USA.

Vandeghinste, V., & Pan, Y. (2004). Sentence compression for automated subtitling: A hybrid approach. In Marie-Francine Moens, S. S. (Ed.), *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pp. 89–95, Barcelona, Spain.

Williams, H. P. (1999). *Model Building in Mathematical Programming* (4th edition). Wiley.

Winston, W. L., & Venkataramanan, M. (2003). *Introduction to Mathematical Programming: Applications and Algorithms* (4th edition). Duxbury.

Zajic, D., Door, B. J., Lin, J., & Schwartz, R. (2007). Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing Management Special Issue on Summarization*, *43*(6), 1549–1570.