A Generic Sentence Trimmer with CRFs

Tadashi Nomoto
National Institute of Japanese Literature
10-3, Midori Tachikawa
Tokyo, 190-0014, Japan
nomoto@acm.org

Abstract

The paper presents a novel sentence trimmer in Japanese, which combines a non-statistical yet generic tree generation model and Conditional Random Fields (CRFs), to address improving the grammaticality of compression while retaining its relevance. Experiments found that the present approach outperforms in grammaticality and in relevance a dependency-centric approach (Oguro et al., 2000; Morooka et al., 2004; Yamagata et al., 2006; Fukutomi et al., 2007) – the only line of work in prior literature (on Japanese compression) we are aware of that allows replication and permits a direct comparison.

1 Introduction

For better or worse, much of prior work on sentence compression (Riezler et al., 2003; McDonald, 2006; Turner and Charniak, 2005) turned to a single corpus developed by Knight and Marcu (2002) (K&M, henceforth) for evaluating their approaches.

The K&M corpus is a moderately sized corpus consisting of 1,087 pairs of sentence and compression, which account for about 2% of a Ziff-Davis collection from which it was derived. Despite its limited scale, prior work in sentence compression relied heavily on this particular corpus for establishing results (Turner and Charniak, 2005; McDonald, 2006; Clarke and Lapata, 2006; Galley and McKeown, 2007). It was not until recently that researchers started to turn attention to an alternative approach which does not require supervised data (Turner and Charniak, 2005).

Our approach is broadly in line with prior work (Jing, 2000; Dorr et al., 2003; Riezler et al., 2003; Clarke and Lapata, 2006), in that we make use of some form of syntactic knowledge to constrain compressions we generate. What sets this work apart from them, however, is a novel use we make of Conditional Random Fields (CRFs) to select among possible compressions (Lafferty et al., 2001; Sutton and McCallum, 2006). An obvious benefit of using CRFs for sentence compression is that the model provides a general (and principled) probabilistic framework which permits information from various sources to be integrated towards compressing sentence, a property K&M do not share.

Nonetheless, there is some cost that comes with the straightforward use of CRFs as a discriminative classifier in sentence compression; its outputs are often ungrammatical and it allows no control over the length of compression they generates (Nomoto, 2007). We tackle the issues by harnessing CRFs with what we might call dependency truncation, whose goal is to restrict CRFs to working with candidates that conform to the grammar.

Thus, unlike McDonald (2006), Clarke and Lapata (2006) and Cohn and Lapata (2007), we do not insist on finding a globally optimal solution in the space of 2^n possible compressions for an n word long sentence. Rather we insist on finding a most plausible compression among those that are explicitly warranted by the grammar.

Later in the paper, we will introduce an approach called the 'Dependency Path Model' (DPM) from the previous literature (Section 4), which purports to provide a robust framework for sentence compression in Japanese. We will look at how the present approach compares with that of DPM in Section 6.

2 A Sentence Trimmer with CRFs

Our idea on how to make CRFs comply with grammar is quite simple: we focus on only those label sequences that are associated with grammatically correct compressions, by making CRFs look at only those that comply with some grammatical constraints G, and ignore others, regardless of how probable they are.[1] But how do we find compressions that are grammatical? To address the issue, rather than resort to statistical generation models as in the previous literature (Cohn and Lapata, 2007; Galley and McKeown, 2007), we pursue a particular rule-based approach we call a 'dependency truncation,' which as we will see, gives us a greater control over the form that compression takes.

Let us denote a set of label assignments for S that satisfy constraints, by $G(S)$.[2] We seek to solve the following,

$$\mathbf{y}^{\star} = \arg \max_{\mathbf{y} \in G(S)} p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}). \tag{2}$$

There would be a number of ways to go about the problem. In the context of sentence compression, a linear programming based approach such as Clarke and Lapata (2006) is certainly one that deserves consideration. In this paper, however, we will explore a much simpler approach which does not require as involved formulation as Clarke and Lapata (2006) do.

We approach the problem *extentionally*, i.e., through generating sentences that are grammatical, or that conform to whatever constraints there are.

[1]Assume as usual that CRFs take the form,

$$p(\mathbf{y}|\mathbf{x}) \propto \exp\left(\sum_{k,j} \lambda_j f_j(y_k, y_{k-1}, \mathbf{x}) + \sum_i \mu_i g_i(x_k, y_k, \mathbf{x})\right) = \exp[\mathbf{w}^\top \mathbf{f}(\mathbf{x},\mathbf{y})] \tag{1}$$

f_j and g_i are 'features' associated with edges and vertices, respectively, and $k \in \mathcal{C}$, where $\mathcal{C}$ denotes a set of cliques in CRFs. λ_j and μ_i are the weights for corresponding features. $\mathbf{w}$ and $\mathbf{f}$ are vector representations of weights and features, respectively (Tasker, 2004).

[2]Note that a sentence compression can be represented as an array of binary labels, one of them marking words to be retained in compression and the other those to be dropped.

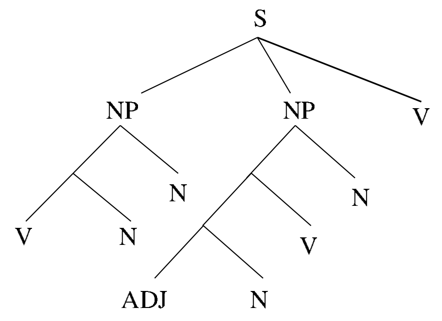

Figure 1: Syntactic structure in Japanese

Consider the following.

(3)

Mushoku-no	*John*	*-ga*	*takai*	*kuruma*
unemployed	John	SBJ	expensive	car
-wo	*kat-ta.*			
ACC	buy PAST			

'John, who is unemployed, bought an expensive car.'

whose grammatically legitimate compressions would include:

(4) (a) *John -ga takai kuruma -wo kat-ta.*
'John bought an expensive car.'

(b) *John -ga kuruma -wo kat-ta.*
'John bought a car.'

(c) *Mushoku-no John -ga kuruma -wo kat-ta.*
'John, who is unemployed, bought a car.

(d) *John -ga kat-ta.*
'John bought.'

(e) *Mushoku-no John -ga kat-ta.*
'John, who is unemployed, bought.'

(f) *Takai kuruma-wo kat-ta.*
' Bought an expensive car.'

(g) *Kuruma-wo kat-ta.*
' Bought a car.'

(h) *Kat-ta.*
' Bought.'

This would give us $G(S)$={a, b, c, d, e, f, g, h}, for the input 3. Whatever choice we make for compression among candidates in $G(S)$, should be grammatical, since they all are. One linguistic feature

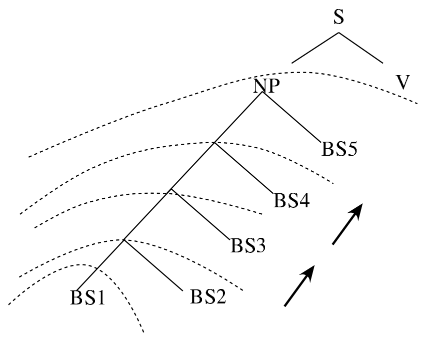

Figure 2: Compressing an NP chunk

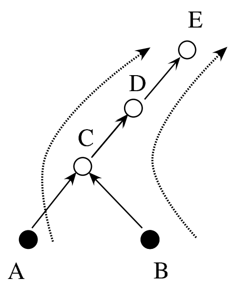

Figure 3: Trimming TDPs

of the Japanese language we need to take into account when generating compressions, is that the sentence, which is free of word order and verb-final, typically takes a left-branching structure as in Figure 1, consisting of an array of morphological units called *bunsetsu* (BS, henceforth). A BS, which we might regard as an inflected form (case marked in the case of nouns) of verb, adjective, and noun, could involve one or more independent linguistic elements such as noun, case particle, but acts as a morphological atom, in that it cannot be torn apart, or partially deleted, without compromising the grammaticality.[3]

Noting that a Japanese sentence typically consists of a sequence of case marked NPs and adjuncts, followed by a main verb at the end (or what would be called 'matrix verb' in linguistics), we seek to compress each of the major chunks in the sentence, leaving untouched the matrix verb, as its removal often leaves the sentence unintelligible. In particular, starting with the leftmost BS in a major constituent, we work up the tree by pruning BSs on our way up, which in general gives rise to grammatically legitimate compressions of various lengths (Figure 2).

More specifically, we take the following steps to construct $G(S)$. Let S = ABCDE. Assume that it has a dependency structure as in Figure 3. We begin by locating terminal nodes, i.e., those which have no incoming edges, depicted as filled circles in Figure 3, and find a dependency (singly linked) path from each terminal node to the root, or a node labeled 'E' here, which would give us two paths p_1 = A-C-D-E and p_2 = B-C-D-E (call them *terminating dependency paths*, or TDPs). Now create a set $\mathcal{T}$ of all trimmings, or suffixes of each TDP, including an empty string:

$$\mathcal{T}(p_1) = \{\text{<A C D E>, <C D E>, <D E>, <E>, <>}\}$$
$$\mathcal{T}(p_2) = \{\text{<B C D E>, <C D E>, <D E>, <E>, <>}\}$$

Then we merge subpaths from the two sets in every possible way, i.e., for any two subpaths $t_1 \in \mathcal{T}(p_1)$ and $t_2 \in \mathcal{T}(p_2)$, we take a union over nodes in t_1 and t_2; Figure 4 shows how this might done. We remove duplicates if any. This would give us $G(S)$={{A B C D E}, {A C D E}, {B C D E}, {C D E}, {D E}, {E}, {}}, a set of compressions over S based on TDPs.

What is interesting about the idea is that creating $G(S)$ does not involve much of anything that is specific to a given language. Indeed this could be done on English as well. Take for instance a sentence at the top of Table 1, which is a slightly modified lead sentence from an article in the *New York Times*. Assume that we have a relevant dependency structure as shown in Figure 5, where we have three TDPs, i.e., one with *southern*, one with *British* and one with *lethal*. Then $G(S)$ would include those listed in Table 1. A major difference from Japanese lies in the direction in which a tree is branching out: right versus left.[4]

Having said this, we need to address some language specific constraints: in Japanese, for instance, we should keep a topic marked NP in compression as its removal often leads to a decreased readability; and also it is grammatically wrong to start any compressed segment with sentence nominalizers such as

[3]Example 3 could be broken into BSs: */ Mushuku -no / John -ga / takai / kuruma -wo / kat-ta /.*

[4]We stand in a marked contrast to previous 'grafting' approaches which more or less rely on an ad-hoc collection of transformation rules to generate candidates (Riezler et al., 2003).

Table 1: Hedge-clipping English

An official was quoted yesterday as accusing Iran of supplying explosive technology used in lethal attacks on British troops in southern Iraq
An official was quoted yesterday as accusing Iran of supplying explosive technology used in lethal attacks on British troops in Iraq
An official was quoted yesterday as accusing Iran of supplying explosive technology used in lethal attacks on British troops
An official was quoted yesterday as accusing Iran of supplying explosive technology used in lethal attacks on troops
An official was quoted yesterday as accusing Iran of supplying explosive technology used in lethal attacks
An official was quoted yesterday as accusing Iran of supplying explosive technology used in attacks
An official was quoted yesterday as accusing Iran of supplying explosive technology
An official was quoted yesterday as accusing Iran of supplying technology

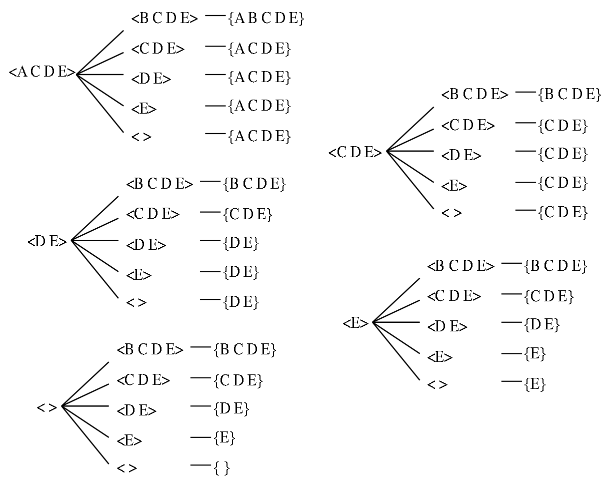

Figure 4: Combining TDP suffixes

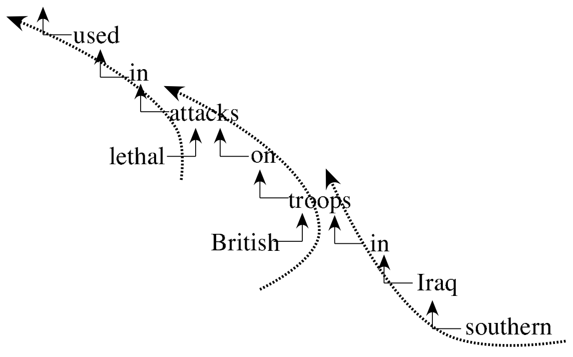

Figure 5: An English dependency structure and TDPs

-koto and *-no*. In English, we should keep a preposition from being left dangling, as in *An official was quoted yesterday as accusing Iran of supplying technology used in*. In any case, we need some extra rules on $G(S)$ to take care of language specific issues (cf. Vandeghinste and Pan (2004) for English). An important point about the dependency truncation is that for most of the time, a compression it generates comes out reasonably grammatical, so the number of 'extras' should be small.

Finally, in order for CRFs to work with the compressions, we need to translate them into a sequence of binary labels, which involves labeling an element token, *bunsetsu* or a word, with some label, e.g., 0 for 'remove' and 1 for 'retain,' as in Figure 6.

Consider following compressions $\mathbf{y}_1$ to $\mathbf{y}_4$ for $\mathbf{x} = \beta_1\beta_2\beta_3\beta_4\beta_5\beta_6$. β_i denotes a *bunsetsu* (BS). '0' marks a BS to be removed and '1' that to be retained.

	β_1	β_2	β_3	β_4	β_5	β_6
$\mathbf{y}_1$	0	1	1	1	1	1
$\mathbf{y}_2$	0	0	1	1	1	1
$\mathbf{y}_3$	0	0	0	0	0	1
$\mathbf{y}_4$	0	0	1	0	0	0

Assume that $G(S) = \{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3\}$. Because $\mathbf{y}_4$ is not part of $G(S)$, it is not considered a candidate for a compression for $\mathbf{y}$, even if its likelihood may exceed those of others in $G(S)$. We note that the approach here does not rely on so much of CRFs as a discriminative classifier as CRFs as a strategy for ranking among a limited set of label sequences which correspond to syntactically plausible simplifications of input sentence.

Furthermore, we could dictate the length of compression by putbting an additional constraint on out-

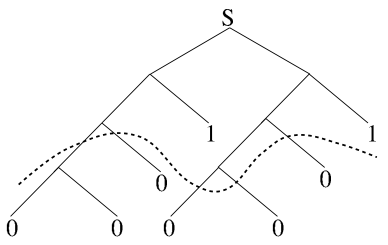

Figure 6: Compression in binary representation.

put, as in:

$$\mathbf{y}^\star = \arg\max_{\mathbf{y} \in G'(S)} p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}), \tag{5}$$

where $G'(S) = \{\mathbf{y} : \mathbf{y} \in G(S), R(\mathbf{y},\mathbf{x}) = r\}$. $R(\mathbf{y},\mathbf{x})$ denotes a compression rate r for which $\mathbf{y}$ is desired, where $r = \frac{\text{\# of 1 in } \mathbf{y}}{\text{length of } \mathbf{x}}$. The constraint forces the trimmer to look for the best solution among candidates that satisfy the constraint, ignoring those that do not.[5]

Another point to note is that $G(S)$ is finite and relatively small $-$ it was found, for our domain, $G(S)$ usually runs somewhere between a few hundred and ten thousand in length $-$ so in practice it suffices that we visit each compression in $G(S)$, and select one that gives the maximum value for the objective function. We will have more to say about the size of the search space in Section 6.

3 Features in CRFs

We use an array of features in CRFs which are either derived or borrowed from the taxonomy that a Japanese tokenizer called JUMAN and KNP,[6] a Japanese dependency parser (aka Kurohashi-Nagao Parser), make use of in characterizing the output they produce: both JUMAN and KNP are part of the compression model we build.

Features come in three varieties: semantic, morphological and syntactic. Semantic features are used for classifying entities into semantic types such as name of person, organization, or place, while syntactic features characterize the kinds of dependency relations that hold among BSs such as whether a BS is of the type that combines with the verb (*renyou*), or of the type that combines with the noun (*rentai*), etc.

A morphological feature could be thought of as something that broadly corresponds to an English POS, marking for some syntactic or morphological category such as noun, verb, numeral, etc. Also we included ngram features to encode the lexical context in which a given morpheme appears. Thus we might have something like: for some words (morphemes) w_1, w_2, and w_3, $f_{w_1 \cdot w_2}(w_3) = 1$ if w_3 is preceded by w_1, w_2; otherwise, 0. In addition, we make use of an IR-related feature, whose job is to indicate whether a given morpheme in the input appears in the title of an associated article. The motivation for the feature is obviously to identify concepts relevant to, or unique to the associated article. Also included was a feature on tfidf, to mark words that are conceptually more important than others. The number of features came to around 80,000 for the corpus we used in the experiment.

4 The Dependency Path Model

In what follows, we will describe somewhat in detail a prior approach to sentence compression in Japanese which we call the "dependency path model," or DPM. DPM was first introduced in (Oguro et al., 2000), later explored by a number of people (Morooka et al., 2004; Yamagata et al., 2006; Fukutomi et al., 2007).[7]

DPM has the form:

$$h(\mathbf{y}) = \alpha f(\mathbf{y}) + (1-\alpha) g(\mathbf{y}), \tag{6}$$

where $\mathbf{y} = \beta_0, \beta_1, \ldots, \beta_{n-1}$, i.e., a compression consisting of any number of *bunsetsu*'s, or phrase-like elements. $f(\cdot)$ measures the relevance of content in $\mathbf{y}$; and $g(\cdot)$ the fluency of text. α is to provide a way of weighing up contributions from each component.

We further define:

$$f(\mathbf{y}) = \sum_{i=0}^{n-1} q(\beta_i), \tag{7}$$

[5]It is worth noting that the present approach can be recast into one based on 'constraint relaxation' (Tromble and Eisner, 2006).

[6]http://nlp.kuee.kyoto-u.ac.jp/nl-resource/top-e.html

[7]Kikuchi et al. (2003) explore an approach similar to DPM.

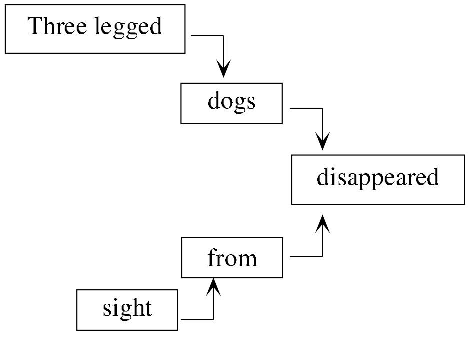

Figure 7: A dependency structure

and

$$g(\mathbf{y}) = \max_{s} \sum_{i=0}^{n-2} p(\beta_i, \beta_{s(i)}). \quad (8)$$

$q(\cdot)$ is meant to quantify how worthy of inclusion in compression, a given *bunsetsu* is; and $p(\beta_i, \beta_j)$ represents the connectivity strength of dependency relation between β_i and β_j. $s(\cdot)$ is a linking function that associates with a *bunsetsu* any one of those that follows it. $g(\mathbf{y})$ thus represents a set of linked edges that, if combined, give the largest probability for $\mathbf{y}$.

Dependency path length (DL) refers to the number of (singly linked) dependency relations (or edges) that span two *bunsetsu*'s. Consider the dependency tree in Figure 7, which corresponds to a somewhat contrived sentence '*Three-legged dogs disappeared from sight*.' Take an English word for a *bunsetsu* here. We have

$$\begin{aligned} &\text{DL(three-legged, dogs)} = 1 \\ &\text{DL(three-legged, disappeared)} = 2 \\ &\text{DL(three-legged, from)} = \infty \\ &\text{DL(three-legged, sight)} = \infty \end{aligned}$$

Since *dogs* is one edge away from *three-legged*, DL for them is 1; and we have DL of two for *three-legged* and *disappeared*, as we need to cross two edges in the direction of arrow to get from the former to the latter. In case there is no path between words as in the last two cases above, we take the DL to be infinite.

DPM takes a dependency tree to be a set of linked edges. Each edge is expressed as a triple $< C_s(\beta_i), C_e(\beta_j), \text{DL}(\beta_i, \beta_j) >$, where β_i and β_j represent *bunsestu*'s that the edge spans. $C_s(\beta)$ denotes the class of a *bunsetsu* where the edge starts and $C_e(\beta)$ that of a *bunsetsu* where the edge ends. What we mean by 'class of *bunsetsu*' is some sort of a classificatory scheme that concerns linguistic characteristics of *bunsetsu*, such as a part-of-speech of the head, whether it has an inflection, and if it does, what type of inflection it has, etc. Moreover, DPM uses two separate classificatory schemes for $C_s(\beta)$ and $C_e(\beta)$.

In DPM, we define the connectivity strength p by:

$$p(\beta_i, \beta_j) = \begin{cases} \log S(t) & \text{if DL}(\beta_i, \beta_j) \neq \infty \\ -\infty & \text{otherwise} \end{cases} \quad (9)$$

where $t =< C_s(\beta_i), C_e(\beta_j), \text{DL}(\beta_i, \beta_j) >$, and $S(t)$ is the probability of t occurring in a compression, which is given by:

$$S(t) = \frac{\text{\# of } t\text{'s found in compressions}}{\text{\# of triples found in the training data}} \quad (10)$$

We complete the DPM formulation with:

$$q(\beta) = \log p_c(\beta) + \text{tfidf}(\beta) \quad (11)$$

$p_c(\beta)$ denotes the probability of having *bunsetsu* β in compression, calculated analogously to Eq. 10,[8] and tfidf(β) obviously denotes the tfidf value of β.

In DPM, a compression of a given sentence can be obtained by finding $\arg\max_{\mathbf{y}} h(\mathbf{y})$, where $\mathbf{y}$ ranges over possible candidate compressions of a particular length one may derive from that sentence. In the experiment described later, we set $\alpha = 0.1$ for DPM, following Morooka et al. (2004), who found the best performance with that setting for α.

5 Evaluation Setup

We created a corpus of sentence summaries based on email news bulletins we had received over five to six months from an on-line news provider called Nikkei Net, which mostly deals with finance and politics.[9] Each bulletin consists of six to seven news briefs, each with a few sentences. Since a news brief contains nothing to indicate what its longer version

[8]DPM puts *bunsetsu*'s into some groups based on linguistic features associated with them, and uses the statistics of the groups for p_c rather than that of *bunsetsu*'s that actually appear in text.

[9]http://www.nikkei.co.jp

Table 2: The rating scale on fluency

RATING	EXPLANATION
1	makes no sense
2	only partially intelligible/grammatical
3	makes sense; seriously flawed in grammar
4	makes good sense; only slightly flawed in grammar
5	makes perfect sense; no grammar flaws

Table 3: The rating scale on content overlap

RATING	EXPLANATION
1	no overlap with reference
2	poor or marginal overlap w. ref.
3	moderate overlap w. ref.
4	significant overlap w. ref.
5	perfect overlap w. ref.

might look like, we manually searched the news site for a full-length article that might reasonably be considered a long version of that brief.

We extracted lead sentences both from the brief and from its source article, and aligned them, using what is known as the Smith-Waterman algorithm (Smith and Waterman, 1981), which produced 1,401 pairs of summary and source sentence.[10] For the ease of reference, we call the corpus so produced 'NICOM' for the rest of the paper. A part of our system makes use of a modeling toolkit called GRMM (Sutton et al., 2004; Sutton, 2006). Throughout the experiments, we call our approach 'Generic Sentence Trimmer' or GST.

6 Results and Discussion

We ran DPM and GST on NICOM in the 10-fold cross validation format where we break the data into 10 blocks, use 9 of them for training and test on the remaining block. In addition, we ran the test at three different compression rates, 50%, 60% and 70%, to learn how they affect the way the models perform. This means that for each input sentence in NICOM, we have three versions of its compression created, corresponding to a particular rate at which the sentence is compressed. We call a set of compressions so generated 'NICOM-g.'

In order to evaluate the quality of outputs GST and DPM generate, we asked 6 people, all Japanese natives, to make an intuitive judgment on how each compression fares in fluency and relevance to gold standards (created by humans), on a scale of 1 to 5. To this end, we conducted evaluation in two separate formats; one concerns fluency and the other relevance. The fluency test consisted of a set of compressions which we created by randomly selecting 200 of them from NICOM-g, for each model at compression rates 50%, 60%, and 70%; thus we have 200 samples for each model and each compression rate.[11] The total number of test compressions came to 1,200.

The relevance test, on the other hand, consisted of *paired* compressions along with the associated gold standard compressions. Each pair contains compressions both from DPM and from GST at a given compression rate. We randomly picked 200 of them from NICOM-g, at each compression rate, and asked the participants to make a subjective judgment on how much of the content in a compression semantically overlap with that of the gold standard, on a scale of 1 to 5 (Table 3). Also included in the survey are 200 gold standard compressions, to get some idea of how fluent "ideal" compressions are, compared to those generated by machine.

Tables 4 and 5 summarize the results. Table 4 looks at the fluency of compressions generated by each of the models; Table 5 looks at how much of the content in reference is retained in compressions. In either table, CR stands for compression rate. All the results are averaged over samples.

We find in Table 4 a clear superiority of GST over DPM at every compression rate examined, with fluency improved by as much as 60% at 60%. However, GST fell short of what human compressions achieved in fluency – an issue we need to address

[10]The Smith-Waterman algorithm aims at finding a best match between two sequences which may include gaps, such as A-C-D-E and A-B-C-D-E. The algorithm is based on an idea rather akin to dynamic programming.

[11]As stated elsewhere, by compression rate, we mean $r = \frac{\text{\# of 1 in } \mathbf{y}}{\text{length of } \mathbf{x}}$.

Table 4: Fluency (Average)

MODEL/CR	50%	60%	70%
GST	**3.430**	**3.820**	**3.810**
DPM	2.222	2.372	2.660
Human	–	4.45	–

Table 5: Semantic (Content) Overlap (Average)

MODEL/CR	50%	60%	70%
GST	**2.720**	**3.181**	**3.405**
DPM	2.210	2.548	2.890

in the future. Since the average CR of gold standard compressions was 60%, we report their fluency at that rate only.

Table 5 shows the results in relevance of content. Again GST marks a superior performance over DPM, beating it at every compression rate. It is interesting to observe that GST manages to do well in the semantic overlap, despite the cutback on the search space we forced on GST.

As for fluency, we suspect that the superior performance of GST is largely due to the dependency truncation the model is equipped with; and its performance in content overlap owes a lot to CRFs. However, just how much improvement GST achieved over regular CRFs (with no truncation) in fluency and in relevance is something that remains to be seen, as the latter do not allow for variable length compression, which prohibits a straightforward comparison between the two kinds of models.

We conclude the section with a few words on the size of $|G(S)|$, i.e., the number of candidates generated per run of compression with GST.

Figure 8 shows the distribution of the numbers of candidates generated per compression, which looks like the familiar scale-free power curve. Over 99% of the time, the number of candidates or $|G(S)|$ is found to be less than 500.

7 Conclusions

This paper introduced a novel approach to sentence compression in Japanese, which combines a syntactically motivated generation model and CRFs, in order to address fluency and relevance of compressions we generate. What distinguishes this work from prior research is its overt withdrawal from a search for global optima to a search for local optima that comply with grammar.

We believe that our idea was empirically borne out, as the experiments found that our approach outperforms, by a large margin, a previously known method called DPM, which employs a global search strategy. The results on semantic overlap indicates that the narrowing down of compressions we search obviously does not harm their relevance to references.

An interesting future exercise would be to explore whether it is feasible to rewrite Eq. 5 as a linear integer program. If it is, the whole scheme of ours would fall under what is known as 'Linear Programming CRFs' (Tasker, 2004; Roth and Yih, 2005). What remains to be seen, however, is whether GST is transferrable to languages other than Japanese, notably, English. The answer is likely to be yes, but details have yet to be worked out.

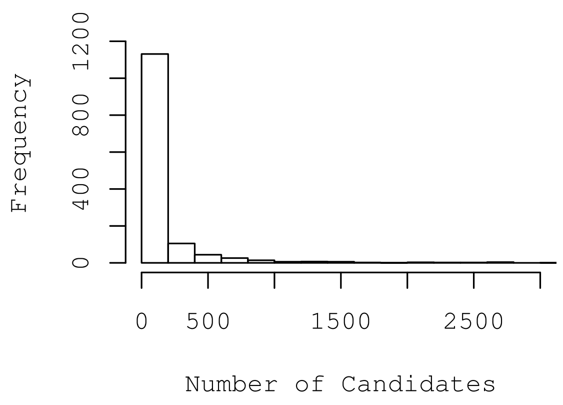

Figure 8: The distribution of $|G(S)|$

References

James Clarke and Mirella Lapata. 2006. Constraint-based sentence compression: An integer programming approach. In *Proceedings of the COLING/ACL 2006*, pages 144–151.

Trevor Cohn and Mirella Lapata. 2007. Large margin synchronous generation and its application to sentence compression. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 73–82, Prague, June.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generataion. In *Proceedings of the HLT-NAACL Text Summarization Workshop and Document Understanding Conderence (DUC03)*, pages 1–8, Edmonton, Canada.

Satoshi Fukutomi, Kazuyuki Takagi, and Kazuhiko Ozeki. 2007. Japanese Sentence Compression using Probabilistic Approach. In *Proceedings of the 13th Annual Meeting of the Association for Natural Language Processing Japan*.

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

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, pages 310–315.

Tomonori Kikuchi, Sadaoki Furui, and Chiori Hori. 2003. Two-stage automatic speech summarization by sentence extraction and compaction. In *Proceedings of ICASSP 2003*.

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

John Lafferty, Andrew MacCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of the 11th Conference of EACL*, pages 297–304.

Yuhei Morooka, Makoto Esaki, Kazuyuki Takagi, and Kazuhiko Ozeki. 2004. Automatic summarization of news articles using sentence compaction and extraction. In *Proceedings of the 10th Annual Meeting of Natural Language Processing*, pages 436–439, March. (In Japanese).

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

Rei Oguro, Kazuhiko Ozeki, Yujie Zhang, and Kazuyuki Takagi. 2000. An efficient algorithm for Japanese sentence compaction based on phrase importance and inter-phrase dependency. In *Proceedings of TSD 2000 (Lecture Notes in Artificial Intelligence 1902,Springer-Verlag)*, pages 65–81, Brno, Czech Republic.

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 2003*, pages 118–125, Edmonton.

Dan Roth and Wen-tau Yih. 2005. Integer linear programming inference for conditional random fields. In *Proceedings of the 22nd International Conference on Machine Learning (ICML 05)*.

T. F. Smith and M. S. Waterman. 1981. Identification of common molecular subsequence. *Journal of Molecular Biology*, 147:195–197.

Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press. To appear.

Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic labeling and segmenting sequence data. In *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada.

Charles Sutton. 2006. GRMM: A graphical models toolkit. http://mallet.cs.umass.edu.

Ben Tasker. 2004. *Learning Structured Prediction Models: A Large Margin Approach*. Ph.D. thesis, Stanford University.

Roy W. Tromble and Jason Eisner. 2006. A fast finite-state relaxation method for enforcing global constraint on sequence decoding. In *Proceeings of the NAACL*, pages 423–430.

Jenie Turner and Eugen Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 290–297, Ann Arbor, June.

Vincent Vandeghinste and Yi Pan. 2004. Sentence compression for automatic subtitling: A hybrid approach. In *Proceedings of the ACL workshop on Text Summarization*, Barcelona.

Kiwamu Yamagata, Satoshi Fukutomi, Kazuyuki Takagi, and Kzauhiko Ozeki. 2006. Sentence compression using statistical information about dependency path length. In *Proceedings of TSD 2006 (Lecture Notes in Computer Science, Vol. 4188/2006)*, pages 127–134, Brno, Czech Republic.