Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Using First-Order Logic to Compress Sentences

Minlie Huang, Xing Shi, Feng Jin, Xiaoyan Zhu
State Key Laboratory of Intelligent Technology and Systems,
Tsinghua National Laboratory for Information Science and Technology,
Department of Computer Science and Technology,
Tsinghua University, Beijing 100084, PR China.
{aihuang,zxy-dcs}@tsinghua.edu.cn, {shixing19910105,jinfengfeng}@gmail.com

Abstract

Sentence compression is one of the most challenging tasks in natural language processing, which may be of increasing interest to many applications such as abstractive summarization and text simplification for mobile devices. In this paper, we present a novel sentence compression model based on first-order logic, using Markov Logic Network. Sentence compression is formulated as a word/phrase deletion problem in this model. By taking advantage of first-order logic, the proposed method is able to incorporate local linguistic features and to capture global dependencies between word deletion operations. Experiments on both written and spoken corpora show that our approach produces competitive performance against the state-of-the-art methods in terms of manual evaluation measures such as importance, grammaticality, and overall quality.

Introduction

Text-to-text generation methods have received much attention for many natural language processing applications including text summarisation, question answering, and machine translation. As an example, questions in question answering are often paraphrased in order to achieve more flexible matching with potential answers (Lin and Pantel 2001).

Sentence compression is perhaps one of the most popular text-to-text generation methods. The aim is to produce a condensed version of the original sentence while retaining the most important information and making the compressed sentence grammatical (Jing 2000). Sentence compression can help text summarization systems to reduce redundancy in generated summaries, and may be a feasible way to generate non-extractive summaries (Knight and Marcu 2002). In addition, it benefits a variety of NLP applications such as subtitle generation and text generation for mobile devices (Filippova and Strube 2008; Clarke and Lapata 2008).

A variety of approaches have been proposed for sentence compression. Some sophisticated compression models such as tree-to-tree transduction models (Cohn and Lapata 2008) (Cohn and Lapata 2009), were able to perform substitution, reordering, inserting, and deletion operations in the compression process, while most other models address a simpler problem that formulates the task as a word/phase deletion problem, i.e., only deletion operation is considered in compression. To name a few, Knight and Marcu (2002) presented a decision tree based model and a noisy channel method. McDonald (2006) employed a large margin learning and a rich set of features to shorten sentences. Conditional Random Field (Nomoto 2009) and Support Vector Regression (Galanis and Androutsopoulos 2010) have also been used in prior studies. The deletion problem was also formulated as an Integer Linear Programming (ILP) problem (Clarke and Lapata 2008; Filippova and Strube 2008), which addressed global dependency in deleting local words.

In this paper we present a new deletion-based compression model using first-order logic and probabilistic model. Compression is implemented through word/phrase deletion operation which is represented by logic formula. Local linguistic features such as tokens, part-of-speech tags, and dependency relations are represented by local logic formulas, and the dependency between multiple deletion operations is modeled as global logic formulas to ensure the grammaticality of compressed sentences. The probabilistic model, Markov Logic Network (MLN) (Richardson and Domingos 2006), is adopted for learning and prediction. MLN is a statistical relational learning framework which has been widely applied in natural language processing such as semantic role labeling (Meza-Ruiz and Riedel 2009), temporal relation identification (Yoshikawa et al. 2009), and question classification (Bu et al. 2010).

To the best of our knowledge, our approach is the first work of using first-order logic for sentence compression. This paper is also a substantial extension to our Chinese version (Jin, Huang, and Zhu 2011). It demonstrates several advantages. First, it is a supervised method and therefore can easily incorporate rich features, in a form of logic formula. Second, it offers us much flexibilities to capture local linguistic features and global dependencies between word deletion operations by designing various logic formulas. Each logic formula can be viewed as a compression rule, or a compression template that instantiates a set of similar compression rules. Various formulas are combined in a probabilistic framework (MLN) which is able to model soft constraints in deletion operations. Third, our approach benefits from the off-the-shelf Markov logic engines and thus the amount of engineering work in system construction has been largely

reduced.

The rest of the paper is organized as follows: in the next section, we present a brief introduction to Markov Logic Network. Then, we introduce our sentence compression model, with local formulas that reflect local linguistic features for compression and global formulas that capture the dependency between deletion operations. Subsequently, we describe the experiments and evaluation results by comparing our method to the state-of-the-art. Finally, we summarize the work of this paper.

Markov Logic Network

Markov Logic Network (MLN) is a statistical relational learning framework that combines first-order Logic and Markov Networks. MLN is a framework to soften the hard (with a probability of either 1 or 0) logic formulas that describe a first-order knowledge base. And formulas can be violated at the cost of some penalty. From an alternative point of view, it is an expressive template language that uses first-order Logic to instantiate Markov Networks of repetitive structure (Meza-Ruiz and Riedel 2009).

As an example, assume we are going to compress the sentence *You are absolutely right* using MLN. Firstly, we introduce a set of logic *predicates* such as $word(p, w)$ and $delete(p)$ which respectively mean that the token at position p is word w and that the token at token at p should be removed. Then, a set of weighted formulas are defined to describe a distribution over sets of *ground atoms* of these predicates. As mentioned previously, a ground atom is an predicate whose variables are bound to constants. A set of ground atoms, termed a possible *world*, represents a compression solution for a sentence. From the point of view of probability, worlds that represent correct compression solutions should receive higher probabilities than worlds that produce incorrect compressions. For example, the model should assign a higher probability to the following world:

$$\{word(1, you), word(2, are), word(3, absolutely), word(4, right), delete(3)\},$$

(The corresponding compressed result is: "You are right") while it would assign a lower probability to this world:

$$\{word(1, you), word(2, are), word(3, absolutely), word(4, right), delete(4)\}.$$

(The corresponding compressed result is: "You are absolutely")

More formally, a Markov logic network L is a set of pairs (ϕ, w), where ϕ is a formula in first-order logic and w is a weight for the formula, a real number. Together with a finite set of constants C = $\{c_1, c_2, ..., c_{|C|}\}$, it defines a Markov network $M_{L,C}$ as follows (Richardson and Domingos 2006):

1. $M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground predicate is true, and 0 otherwise.
2. $M_{L,C}$ contains one feature for each possible grounding of each formula ϕ in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w associated with ϕ in L.

According to the definition, the graphical structure of $M_{L,C}$ is built as follows: there is an edge between two nodes of $M_{L,C}$ iff the corresponding ground atoms appear together in at least one grounding of one formula in L. Thus, the atoms in each ground formula form a (not necessarily maximal) clique in the graph of $M_{L,C}$.

Then, a Markov Logic Network (MLN) L defines a distribution over possible worlds as follows (Richardson and Domingos 2006; Meza-Ruiz and Riedel 2009):

$$p(\mathbf{y}) = \frac{1}{Z} \exp\left(\sum_{(\phi,w)\in L} w \sum_{\mathbf{c}\in C^{\phi}} f_{\mathbf{c}}^{\phi}(\mathbf{y}) \right) \quad (1)$$

where $\mathbf{y}$ is a possible world; each $\mathbf{c}$ is a binding of free variables in ϕ to constants; $f_{\mathbf{c}}^{\phi}(\mathbf{y})$ is a binary feature function that returns 1 if a true value is obtained in the ground formula in which free variables ($\mathbf{y}$) of ϕ was replaced by constants in $\mathbf{c}$, and 0 otherwise; C^{ϕ} is all possible bindings of variables to constants, and Z is a normalization factor.

MLN weights can be learned generatively (Richardson and Domingos 2006) or discriminatively (Singla and Domingos 2005). There are several MLN learning packages available such as Alchemy [1], PyMLNs [2], thebeast [3], and so on. In our experiment, we leveraged thebeast Markov logic engine for learning and prediction.

The Proposed Model

As stated before, our purpose is to compress a sentence by removing unimportant words. For this task, there is a single hidden predicate $delete(p)$ indicating that the token at position p is deleted in the compressed sentence. In addition, we also extract a set of observed predicates from the training dataset, as listed in Table 1.

In our setting, a formula is considered local if it includes exactly one hidden ground atom for the deletion operation of a single token. In contrast, a global formula incorporates multiple hidden ground atoms. We formulate both local and global formulas to model sentence compression. In constructing these formulas, Stanford parser[4] is used for POS tagging and for generating dependency relations.

Local Formulas

The local formulas represent the features of a token to decide whether it is syntactically critical or conveys important information. The first two formulas address the influence of word form and POS tag features. And this reflects the fact that certain words such as *determiner*, *adjectives* or *adverbs*

[1] http://alchemy.cs.washington.edu/

[2] http://www9-old.in.tum.de/people/jain/mlns/

[3] http://thebeast.googlecode.com/

[4] http://nlp.stanford.edu/software/lex-parser.shtml

$word(i, w)$	Token i has word w
$pos(i, t)$	Token i has POS tag t
$verb(i)$	Token i is a verb
$keep(i)$	Token i is kept in compression
$dep(h, m, d)$	Token h and token m are governor and dependent of dependency relation d, respectively
$tf(i, v)$	Term frequency of token i is v
$idf(i, v)$	Inverse document frequency of token i is v
$cldepth(i, n)$	The nesting depth of the clause that contains token i is n
$depstrength(h, m, f)$	The strength of dependency relation between governor h and dependent m is f

Table 1: Observable predicates

are more likely to be deleted.

$$word(i, +w) \Rightarrow delete(i) \tag{2}$$

$$pos(i, +t) \Rightarrow delete(i) \tag{3}$$

The + notation in a formula indicates that the MLN engine[5] should produce a separate formula and learn a separate weight for each constant of the logic variable. For example, formula $word(2, are) \Rightarrow delete(2)$ and $word(2, right) \Rightarrow delete(2)$ may have different weights as inferred by Formula (2). Similarly, $pos(2, det) \Rightarrow delete(2)$ and $pos(2, pobj) \Rightarrow delete(2)$ have different weights.

Syntactic features are helpful in determining which parts of a sentence may be removed. Generally speaking, it is risky to remove a main verb, and its direct subjective or direct objective, but a modifier or a supplement can be deleted more safely. As an example, the dependency relation $rcmod(book - 4, bought - 6)$ in sentence "*I saw the book you bought*" indicates that the relative clausal modifier (*rcmod*) *bought* of *book* may be dropped. The following two formulas take into account such dependency relation between a governor and a dependent.

$$\begin{gathered} word(h, w_1) \wedge word(m, w_2) \wedge dep(h, m, +d) \\ \Rightarrow delete(m) \end{gathered} \tag{4}$$

$$\begin{gathered} pos(h, +t_1) \wedge pos(m, +t_2) \wedge dep(h, m, +d) \\ \Rightarrow delete(m) \end{gathered} \tag{5}$$

The conditional probability $p(d|h)$ of a dependency relation d given a head word h measures the strength of the relation and prevents us from breaking tight dependencies in a sentence (Filippova and Strube 2008). This concern is handled by the following formula where f is $p(d|h)$ when the dependency between h and m is d.

$$\begin{gathered} word(h, w_1) \wedge word(m, w_2) \wedge \\ depstrength(h, m, +f) \Rightarrow delete(m) \end{gathered} \tag{6}$$

In practice, we are usually required to compress a group of related sentences (e.g., a document) rather than a single isolated one. So we introduced formulas (7) and (8) to incorporate Term Frequency and Inverse Document Frequency of words to measure the salience of information they convey.

$$word(i, w) \wedge tf(i, +v) \Rightarrow delete(i) \tag{7}$$

$$word(i, w) \wedge idf(i, +v) \Rightarrow delete(i) \tag{8}$$

The last local formula determines the salience of a word using the depth of the clause it occurs. Intuitively, in a complex sentence a more deeply nested clause tends to convey more important information (Clarke and Lapata 2008).

$$word(i, w) \wedge cldepth(i, +d) \Rightarrow delete(i) \tag{9}$$

Global Formulas

While the local formulas deal with the deletion of a single token, each global formula is designed to handle the deletion of multiple tokens. Or, in other word, global formulas deal with the dependency among multiple deletion operations. The global formulas defined here attempt to keeping the compressed sentences grammatical, and meanwhile to ensuring a satisfactory compression rate.

For many dependency relations, such as $amod$, det and $advmod$, when the head/governor word is dropped the modifier/dependent word must also be dropped. And when the modifier is kept, the head is also retained. For example in the sentence "*Sam eats red meat and oranges*", the dependency relation between red and meat is $amod(meat, red)$, and when the head *meat* is removed, the adjectival modifier *red* should be removed simultaneously. We have the following formula to address this issue:

$$\begin{gathered} word(h, w_1) \wedge word(m, w_2) \wedge dep(h, m, d) \\ \wedge keephead(d) \wedge delete(h) \Rightarrow delete(m) \end{gathered} \tag{10}$$

where $keephead(d)$ is a predicate that determines whether the dependency relation d should be subject to the head-modifier constraint stated above.

Another grammatical constraint is that for some particular dependency relations the head and modifier words should also be dropped or kept simultaneously. Examples include the neg, $pobj$, and so on. Thus, in sentence "*I sat on the chair*" the word *chair* is $pobj$ of word *on*, and they should be removed or retained together. So we add a formula to model this:

$$\begin{gathered} word(h, w_1) \wedge word(m, w_2) \wedge dep(h, m, d) \\ \wedge simdel(d) \Rightarrow (delete(m) \Leftrightarrow delete(h)) \end{gathered} \tag{11}$$

where the predicate $simdel(d)$ indicates head and modifier should be dropped or retained simultaneously for dependency d.

We also have a formula to force the compressed sentence to contain at least one verb when the original sentence has a main verb. This is implemented by the following formulas:

$$\begin{gathered} word(i, w) => (delete(i) =>!keep(i)) \\ word(i, w) => (!delete(i) => keep(i)) \\ |\{verb(i)\ \&\ keep(i) | all\ i\}| >= 1 \end{gathered} \tag{12}$$

[5] *Thebeast* is used in this paper.

The number of words (len)	The value of α
$len < 5$	0
$5 \leq len < 10$	$len * 0.3$
$10 \leq len < 20$	$len * 0.35$
$20 \leq len < 30$	$len * 0.45$
$len \geq 30$	$len * 0.60$

Table 2: The detailed settings of α. Larger α means removing more words in compression.

The first two formulas assert that $keep(i)$ and $delete(i)$ are mutually exclusive to each other. The third formula asserts that at least one verb is kept during compression.

To improve grammaticality, we also define a set of predicates and global formulas to deal with the deletion of punctuations. We will first exemplify the left and right range of a punctuation that spans. For example, in a coordinate structure, "..., $A_1 A_2, andB$", the second comma spans two words (A_1, A_2), and it's left range is $\{A_1, A_2\}$. For restrictive attributive clauses, such as *"A, which is a ..., has been proved."*, the first comma has both left range and right range, where the right range is $\{which, is, a, ...\}$. The central idea is that if all words that a punctuation spans have been deleted, then the punctuation should also be deleted. This rule applies to both coordinate and appositive structures. For the conciseness of the paper, we will not go into details of this part, while the full version of predicates and global formulas is available at: http://www.qanswers.net/faculty/hml/.

The last consideration is about compression rate. Our assumption is that longer sentences should be compressed more while shorter sentences less. Thus, the formula is imposed to remove at least α words from a sentence. The value of α is determined similar to that in (Filippova and Strube 2008). The details are shown in Table 2.

Experimental Results

Data Preparation

We conducted experiments on both written and spoken corpora[6] adopted from (Clarke and Lapata 2008). The corpora contain source sentences and human-crafted compressions (by deleting words) which are used as gold standard. The written corpus consists of 82 newspaper articles (1,629 sentences) while the spoken corpus contains 50 broadcast stories (1,370 sentences). For experiments on the written corpus, 74 articles containing 1,471 sentences were randomly selected for training, and the left 158 sentences were used for evaluation. As for the spoken corpus, the training set has 1,223 sentences from 45 randomly chosen articles and the left 147 sentences for test.

Both automatic and manual evaluation have been applied in previous studies. In regard to automatic evaluation, sentence compression systems are commonly measured in two metrics. The first one is compression rate measuring the length difference before and after compression, while the second one is the F-score of dependency relations of computer-generated compressions against those of human-crafted compressions (Riezler et al. 2003; Clarke and Lapata 2008). As for manual evaluation, evaluators were asked to assign scores from 1 to 5 in terms of grammaticality and importance respectively for each compressed sentence. In this paper, we introduce another metric, *overall rank*, to measure the overall quality of automatic compression performance among multiple systems.

We compared our method against two publicly available state-of-the-art systems. The first system is T3 (Cohn and Lapata 2009) which learns parse tree transduction rules from a parallel corpus using a large margin method. And the other one is a two-stage approach - SVR-TOKACC-LM (SVTL), proposed by (Galanis and Androutsopoulos 2010). It first generates candidate compression by removing edges from dependency trees using Maximum Entropy and then selects the best candidate using Support Vector Regression.

In our setting, we used Stanford parser and thebeast Markov Logic engine. The language model required by T3 and SVTL is trained using SRILM[7] on the AQUAINT[8] dataset. And the dependency strength $p(d|h)$ is estimated using 30,000 sentences randomly chosen from the AQUAINT corpus. We used the default configuration and conducted experiments in strict compliance with the specifications described in the documentations of T3 and SVTL, respectively.

The detailed setting of thebeast engine is as follows: The model is trained by an online learning algorithm with the MIRA update rule, and the loss function is GlobalNumberOfErrors (default) which counts the number of false positives and false negatives. The inference algorithm is the MAP inference with a cutting plane approach. The initial weights are set to all zeros, and the number of iterations is set to 10 epochs. More detailed manual can be found at http://thebeast.googlecode.com/.

Automatic Evaluation

The automatic evaluation results on the written corpus is presented in Table 3. Our approach has achieved a compression rate of 72.4%, close to that of human-crafted compressions (70.3%), as reported in (Clarke and Lapata 2008). And our approach outperforms SVTL with a better (i.e., producing shorter sentences) compression rate (72.4% vs. 79.9%) and a higher F-score (70.5% vs. 60.7%). Due to quite different compression rates, our method can not be compared with T3 directly. But according to (Galanis and Androutsopoulos 2010), SVTL is comparable with or superior to T3, which is also justified by our manual evaluation in the next subsection.

Table 4 presents the experimental results on the spoken corpus. Our system has a slightly worse compression rate than T3 (75.1% vs. 70.0%), but a much higher F-score (67.2% vs. 50.8%). The system also produces shorter compressions in comparison to SVTL (75.1% vs. 83.5%), and also better F-scores (67.2% vs. 62.5%), which means that our system produces shorter sentences and maintains more important information than SVTL. Compression results by 2

[6] http://jamesclarke.net/research/resources

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

[8] http://www.ldc.upenn.edu/Catalog/docs/LDC2002T31/

System	Compression Rate(%)	F-score(%)
Ours	72.4	70.5
SVTL	79.9	60.7
T3	60.6	43.9

Table 3: Automatic evaluation results on the written corpus.

System	Compression Rate(%)	F-score(%)
Ours	75.1	67.2
SVTL	83.5	62.5
T3	70.0	50.8
Annotator2	80.5	75.3
Annotator3	71.1	71.2

Table 4: Automatic evaluation results on the spoken corpus.

human annotators show that the human annotators perform much better than automatic compression approaches[9].

Manual Evaluation

We performed manual evaluation on both written and spoken corpora. Three subjects were asked to label each compressed sentence in terms of importance, grammaticality, and overall rank, respectively. The subjects were required to label each compressed sentence in a scale of 1-5 stars (scores like 3.5 or 4.5 are also permitted). Importance measures how well the important information in the original sentence has been retained. Grammaticality indicates how good the readability is of the compressed sentences. The subjects were asked to label compressed sentences from four systems, i.e., human-crafted compression (labeled as Human in Table 5), our approach (labeled as Ours), T3, and SVTL. As the labeling task for scoring the compressed sentences is quite subjective and complex[10], we also asked the subjects to determine the partial order between the three automatic compression systems (i.e., our approach, T3, and SVTL) in terms of the overall quality of a compression [11]. This comes up with a new metric, the *overall rank* of a compression system among multiple systems. For example, if our system produces a better compression than T3, and T3 better than SVTL, the overall rank scores for our system, T3, and SVTL are 1.0, 2.0, and 3.0, respectively. If two systems produce equally good compression for a sentence, the overall rank scores for the two systems are the same. A smaller overall rank score means a better performance of a system. The means and standard deviations of importance, grammaticality, and overall rank scores are computed over all test sentences.

During annotation, the identifier of the system from which the compressed sentence was generated is not visible to the subjects to avoid labeling bias, however, the subjects are aware of which compression is crafted by human annotators. The reason for this is, that human-crafted compression can be used as reference to score automatic compressions, since scoring compressed sentences is quite subjective and complex, as mentioned previously.

Written			
System	Importance	Grammaticality	Overall Rank
Human	4.82 ± 0.40	4.97 ± 0.20	N/A
Ours	4.20 ± 0.95	4.50 ± 0.88	1.52 ± 0.72
SVTL	4.08 ± 1.05	4.63 ± 0.77	1.81 ± 0.74
T3	3.48 ± 1.21	4.04 ± 1.16	2.23 ± 0.83
Spoken			
System	Importance	Grammaticality	Overall Rank
Human	4.83 ± 0.43	4.95 ± 0.21	N/A
Ours	4.22 ± 1.01	4.45 ± 0.99	1.51 ± 0.70
SVTL	4.32 ± 0.96	4.71 ± 0.74	1.65 ± 0.72
T3	3.71 ± 1.27	4.14 ± 1.16	2.09 ± 0.84

Table 5: Mean and standard deviations of importance score, grammticality score, and overall rank with the manual evaluation.

The evaluation results are presented in Table 5. Here are some observations: First, human-crafted compression is remarkably better than automatic compression systems. Second, our approach is better than T3 in terms of all metrics (i.e., importance, grammaticality, and overall rank). Third, our approach is comparable with or superior to the two-stage machine learning approach, SVTL. Particularly, our system outperforms T3 and SVTL in terms of overall rank (Note that smaller the score is, the better the compression performance). Note, for this particular labeling task, that the overall rank metric may be more convincible than the importance and grammaticality metrics that require absolute ratings.

Examples and Discussions

We present two examples in Table 6 and Table 7 respectively. In the first example, our model produces perfect compression as the human annotator does. T3 generates wrong compression, while SVTL tends to have a worse compression rate (i.e., the compressed sentence is longer). In the second example, our model is comparable to the human annotator, but T3 has a worse compression rate, and SVTL has poor grammaticality. Two bad examples are also given in Table 8.

By further comparing the compression results, we found that our system has a better balance between compression rate and maintenance of important information than T3 and SVTL. T3 sometimes keeps the modifier while removes the head word, and introduces meaningless constituents that do not occur in the source sentence because it can perform substitution or insert beyond deletion (Cohn and Lapata 2008). As for SVTL, it may remove key constituents such as direct object and negation word. As an example, "*This was not a*

[9]The results by the two annotators in Table 4 are derived from the original corpus. The annotators here are not referred to subjects in the manual evaluation introduced in the next subsection.

[10]This was also demonstrated by the large standard deviations in the evaluation results of Table 5.

[11]We believe that comparison between two sentences (i.e., determining the partial order) may be easier than giving an absolute rating for each individual sentence.

Original	From the bottom of the list of nominees he climbed to the top.
Human	he climbed to the top.
Ours	he climbed to the top.
T3	the list of nominees climbed to the top.
SVTL	From the bottom of the list of nominees he climbed to the top.

Table 6: A good compression example from the written corpus.

Original	We 'll have Steve back next Monday morning to bring us a little bit more information.
Human	We 'll have Steve back Monday morning to bring us more.
Ours	We 'll have Steve back next Monday morning to bring us information.
T3	We 'll have Steve back next Monday morning to bring us a little bit more information.
SVTL	We 'll have Steve back to bring us bit more information.

Table 7: A good compression example from the spoken corpus.

matter of political convenience" is compressed to "*This was a matter of convenience*".

Our compression approach is an elegant combination of rule and statistical learning. Each formula can be viewed as a compression rule (e.g., Formula (10)), or a compression template that instantiates a set of similar compression rules (e.g., the + notation in Formula (2) and (3)). Within the framework of Markov Logic Network, various rules can be combined in a soft, probabilistic way. Therefore, the approach provides us much flexibility to design as many compression rules as possible, in a form of logic formula. Some formulas may be designed to respect local linguistic features, and others to favor grammaticality.

Conclusion and Future Work

We proposed a sentence compression model using first-order logic and Markov Logic Network. The model leverages local and global formulas in first-order logic to deal with deletion operation in compression. Local formulas capture local linguistic features such as part-of-speech tags and dependency relations, and global formulas model the dependency between word deletion operations. In comparison to the state-of-the-art systems, our method exhibits competitive performance on both written and spoken dataset, with both automatic and manual evaluation.

More phrase structure based features will be considered in future work. Such features, represented in formulas, can be easily plugged into the system. Furthermore, in addition to the basic formulas presented here, automatic learning of sophisticated formulas may be obtained through sequence/tree alignment of the source and human-compressed sentences.

Written	
Original	" Many of the things which bring joy to our hearts in the countryside have been destroyed , " said Sir David .
Human	" Many of the things which bring joy in the countryside have been destroyed , " said Sir David .
Ours	said Sir David .
T3	" Many In the things which , " said Sir David .
SVTL	" Many have been destroyed , " said Sir David .
Spoken	
Original	This is a breach that smells a lot–
Human	This breach smells
Ours	a –
T3	This is smells a lot –
SVTL	This is a breach that smells a lot –

Table 8: Two bad compression examples on the written and spoken corpora respectively.

Acknowledgements

This paper was supported by Chinese 973 project under No. 2012CB316301 and National Chinese Science Foundation projects with No. 60803075 and No. 60973104.

References

Bu, F.; Zhu, X.; Hao, Y.; and Zhu, X. 2010. Function-based question classification for general qa. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, 1119–1128. Stroudsburg, PA, USA: Association for Computational Linguistics.

Clarke, J., and Lapata, M. 2008. Global inference for sentence compression an integer linear programming approach. *J. Artif. Int. Res.* 31:399–429.

Cohn, T., and Lapata, M. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, 137–144. Stroudsburg, PA, USA: Association for Computational Linguistics.

Cohn, T., and Lapata, M. 2009. Sentence compression as tree transduction. *J. Artif. Int. Res.* 34:637–674.

Filippova, K., and Strube, M. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, INLG '08, 25–32. Stroudsburg, PA, USA: Association for Computational Linguistics.

Galanis, D., and Androutsopoulos, I. 2010. An extractive supervised two-stage method for sentence compression. In *The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL 2010, 885–893. Stroudsburg, PA, USA: Association for Computational Linguistics.

Jin, F.; Huang, M.; and Zhu, X. 2011. Sentence compression

with a markov logic network. *Journal of Tsinghua University(Science and Technology)* 11:1596–1600.

Jing, H. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the sixth conference on Applied natural language processing*, ANLC '00, 310–315. Stroudsburg, PA, USA: Association for Computational Linguistics.

Knight, K., and Marcu, D. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artif. Intell.* 139:91–107.

Lin, D., and Pantel, P. 2001. Discovery of inference rules for question-answering. *Nat. Lang. Eng.* 7:343–360.

Meza-Ruiz, I., and Riedel, S. 2009. Jointly identifying predicates, arguments and senses using markov logic. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, 155–163. Stroudsburg, PA, USA: Association for Computational Linguistics.

Nomoto, T. 2009. A comparison of model free versus model intensive approaches to sentence compression. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, 391–399. Stroudsburg, PA, USA: Association for Computational Linguistics.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Mach. Learn.* 62:107–136.

Riezler, S.; King, T. H.; Crouch, R.; and Zaenen, A. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, 118–125. Stroudsburg, PA, USA: Association for Computational Linguistics.

Singla, P., and Domingos, P. 2005. Discriminative training of markov logic networks. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 2*, 868–873. AAAI Press.

Yoshikawa, K.; Riedel, S.; Asahara, M.; and Matsumoto, Y. 2009. Jointly identifying temporal relations with markov logic. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, 405–413. Stroudsburg, PA, USA: Association for Computational Linguistics.