Can Syntax Help? Improving an LSTM-based Sentence Compression Model for New Domains

Liangguo Wang[†∗], **Jing Jiang**[∗], **Hai Leong Chieu**[⋆], **Chen Hui Ong**[⋆], **Dandan Song**[†], **Lejian Liao**[†]
chenwangliangguo@bit.edu.cn, jingjiang@smu.edu.sg
{chaileon, ochenhui}@dso.org.sg, {sdd, liaolj}@bit.edu.cn
[†] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
[∗] School of Information Systems, Singapore Management University, Singapore
[⋆] DSO National Laboratories, Singapore

Abstract

In this paper, we study how to improve the domain adaptability of a deletion-based Long Short-Term Memory (LSTM) neural network model for sentence compression. We hypothesize that syntactic information helps in making such models more robust across domains. We propose two major changes to the model: using explicit syntactic features and introducing syntactic constraints through Integer Linear Programming (ILP). Our evaluation shows that the proposed model works better than the original model as well as a traditional non-neural-network-based model in a cross-domain setting.

1 Introduction

Sentence compression is the task of compressing long, verbose sentences into short, concise ones. It can be used as a component of a text summarization system. Figure 1 shows two example input sentences and the compressed sentences written by human. The task has been studied for almost two decades. Early work on this task mostly relies on syntactic information such as constituency-based parse trees to help decide what to prune from a sentence or how to re-write a sentence (Jing, 2000; Knight and Marcu, 2000). Recently, there has been much interest in applying neural network models to solve the problem, where little or no linguistic analysis is performed except for tokenization (Filippova et al., 2015; Rush et al., 2015; Chopra et al., 2016).

Although neural network-based models have achieved good performance on this task recently, they tend to suffer from two problems: (1) They require a large amount of data for training. For example, Filippova et al. (2015) used close to two million sentence pairs to train an LSTM-based sentence compression model. Rush et al. (2015) used about four million title-article pairs from the Gigaword corpus (Napoles et al., 2012) as training data. Although it may be easy to automatically obtain such training data in some domains (e.g., the news domain), for many other domains, it is not possible to obtain such a large amount of training data. (2) These neural network models trained on data from one domain may not work well on out-of-domain data. For example, when we trained a standard neural sequence-to-sequence model[1] on 3.8 million title-article pairs from the Gigaword corpus and applied it to both in-domain data and out-of-domain data, we found that the performance on in-domain data was good but the performance on out-of-domain data could be very

In-domain

Input: *The southern Chinese city of Guangzhou has set up a special zone allowing foreign consulates to build permanent offices and residences and avoid prohibitive local rents, the china daily reported Tuesday.*
Compressed (by human): *Guangzhou opens new consulate area.*
Compressed (by machine): *Guangzhou sets up special zone for foreign consulates.*

Out-of-domain

Input: *Wherever she was, she helped other loyal and flexible wives cope.*
Compressed (by human): *she helped other wives cope.*
Compressed (by machine): *wives and flexible wives*

Figure 1: Examples of in-domain and out-of-domain results by a standard abstractive sequence-to-sequence model trained on the Gigaword corpus. The first input sentence comes from the Gigaword corpus while the second input sentence comes from the written news corpus used by Clarke and Lapata (2008).

[1] http://opennmt.net/

poor. Two example compressed sentences by this trained model are shown in Figure 1 to illustrate the comparison between in-domain and out-of-domain performance.

The two limitations above imply that these neural network-based models may not be good at learning generalizable patterns, or in other words, they tend to overfit the training data. This is not surprising because these models do not explicitly use much syntactic information, which is more general than lexical information.

In this paper, we aim to study how syntactic information can be incorporated into neural network models for sentence compression to improve their domain adaptability. We hope to train a model that performs well on both in-domain and out-of-domain data. To this end, we extend the deletion-based LSTM model for sentence compression by Filippova et al. (2015). Although deletion-based sentence compression is not as flexible as abstractive sentence compression, we chose to work on deletion-based sentence compression for the following reason. Abstractive sentence compression allows new words to be used in a compressed sentence, i.e., words that do not occur in the input sentence. Oftentimes these new words serve as paraphrases of some words or phrases in the source sentence. But to generate such paraphrases, the model needs to have seen them in the training data. Because we are interested in a cross-domain setting, the paraphrases learned in one domain may not work well in another domain if the two domains have very different vocabularies. On the other hand, a deletion-based method does not face such a problem in a cross-domain setting.

Specifically, we propose two major changes to the model by Filippova et al. (2015): (1) We explicitly introduce POS embeddings and dependency relation embeddings into the neural network model. (2) Inspired by a previous method (Clarke and Lapata, 2008), we formulate the final predictions as an Integer Linear Programming problem to incorporate constraints based on syntactic relations between words and expected lengths of the compressed sentences. In addition to the two major changes above, we also use bi-directional LSTM to include contextual information from both directions into the model.

We evaluate our method using around 10,000 sentence pairs released by Filippova et al. (2015) and two other data sets representing out-of-domain data. We test both in-domain and out-of-domain performance. The experimental results showed that our proposed method can achieve competitive performance compared with the original method in the single-domain setting but with much less training data (around 8,000 sentence pairs for training instead of close to two million sentence pairs). In the cross-domain setting, our proposed method can clearly outperform the original method. We also compare our method with a traditional ILP-based method using syntactic structures of sentences but not based on neural networks (Clarke and Lapata, 2008). We find that our method can outperform this baseline for both in-domain and out-of-domain data.

2 Method

In this section, we present our sentence compression method that is aimed at working in a cross-domain setting.

2.1 Problem Definition

Recall that we focus on deletion-based sentence compression. Our problem setup is the same as that by Filippova et al. (2015). Let us use $\boldsymbol{s} = (w_1, w_2, \ldots, w_n)$ to denote an input sentence, which consists of a sequence of words. Here $w_i \in \mathcal{V}$, where $\mathcal{V}$ is the vocabulary. We would like to delete some of the words in $\boldsymbol{s}$ to obtain a compressed sentence that still contains the most important information in $\boldsymbol{s}$. To represent such a compressed sentence, we can use a sequence of binary labels $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$, where $y_i \in \{0, 1\}$. Here $y_i = 0$ indicates that w_i is deleted, and $y_i = 1$ indicates that w_i is retained.

We assume that we have a set of training sentences and their corresponding deletion/retention labels, denoted as $\mathcal{D} = \{(\boldsymbol{s}_j, \boldsymbol{y}_j)\}_{j=1}^{N}$. Our goal is to learn a sequence labeling model from $\mathcal{D}$ so that for any unseen sentence $\boldsymbol{s}$ we can predict its label sequence $\boldsymbol{y}$ and thus compress the sentence.

2.2 Our Base Model

We first introduce our base model, which uses LSTM to perform sequence labeling. This base model is largely based on the model by Filippova et al. (2015) with some differences, which will be explained below.

We assume that each word in the vocabulary has a d-dimensional embedding vector. For input sentence $\boldsymbol{s}$, let us use $(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n)$ to denote the

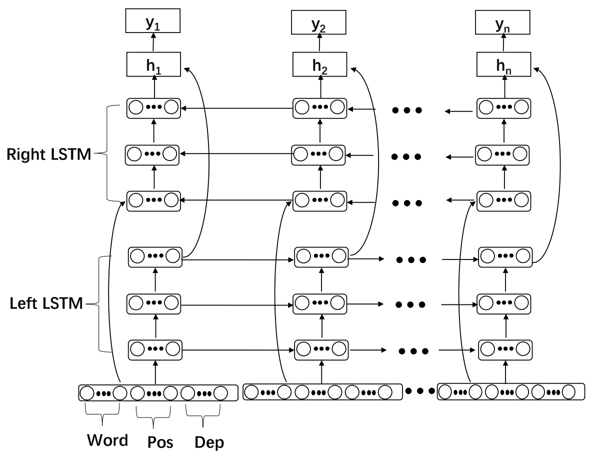

Figure 2: Our three-layered bi-LSTM model. Word embeddings, POS tag embeddings and dependency type embeddings are concatenated in the input layer.

sequence of the word embedding vectors, where $\mathbf{w}_i \in \mathbb{R}^d$. We use a standard bi-directional LSTM model to process these embedding vectors sequentially from both directions to obtain a sequence of hidden vectors $(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)$, where $\mathbf{h}_i \in \mathbb{R}^h$. We omit the details of the bi-LSTM and refer the interested readers to the work by Graves et al. (2013) for further explanation. Following Filippova et al. (2015), our bi-LSTM has three layers, as shown in Figure 2.

We then use the hidden vectors to predict the label sequence. Specifically, label y_i is predicted from $\mathbf{h}_i$ as follows:

$$p(y_i \mid \mathbf{h}_i) = \text{softmax}(\mathbf{W}\mathbf{h}_i + \mathbf{b}), \quad (1)$$

where $\mathbf{W} \in \mathbb{R}^{2 \times h}$ and $\mathbf{b} \in \mathbb{R}^2$ are a weight matrix and a weight vector to be learned.

There are some differences between our base model and the LSTM model by Filippova et al. (2015). (1) Filippova et al. (2015) first encoded the input sentence in its reverse order using the same LSTM before processing the sentence for sequence labeling. (2) Filippova et al. (2015) used only a single-directional LSTM while we use bi-LSTM to capture contextual information from both directions. (3) Although Filippova et al. (2015) did not use any syntactic information in their basic model, they introduced some features based on dependency parse trees in their advanced models. Here we follow their basic model because later we will introduce more explicit syntax-based features. (4) Filippova et al. (2015) combined the predicted y_{i-1} with $\mathbf{w}_i$ to help predict y_i. This adds some dependency between consecutive labels. We do not do this because later we will introduce an ILP layer to introduce dependencies among labels.

2.3 Incorporation of Syntactic Features

Note that in the base model that we presented above, there is no explicit use of any syntactic information such as the POS tags of the words or the parse tree structures of the sentences. Because we believe that syntactic information is important for learning a generalizable model for sentence compression, we would like to introduce syntactic features into our model.

First, we perform part-of-speech tagging on the input sentences. For sentence $\boldsymbol{s}$, let us use $(t_1, t_2, \ldots, t_n)$ to denote the POS tags of the words inside, where $t_i \in \mathcal{T}$ and $\mathcal{T}$ is a POS tag set. We further assume that each $t \in \mathcal{T}$ has an embedding vector (to be learned). Let us use $(\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n)$ ($\mathbf{t}_i \in \mathbb{R}^p$, $p < |\mathcal{T}|$) to denote the sequence of POS embedding vectors of this sentence. We can then simply concatenate $\mathbf{w}_i$ with $\mathbf{t}_i$ as a new vector to be processed by the bi-LSTM model.

Next, we perform dependency parsing on the input sentences. For each word w_i in sentence $\boldsymbol{s}$, let $r_i \in \mathcal{R}$ denote the dependency relation between w_i and its parent word in the sentence, where $\mathcal{R}$ is the set of all dependency relation types. We then assume that each $r \in \mathcal{R}$ has an embedding vector (to be learned). Let $(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n)$ ($\mathbf{r} \in \mathbb{R}^q$, $q < |\mathcal{R}|$) denote corresponding dependency embedding vectors of this sentence. We can also concatenate $\mathbf{w}_i$ with $\mathbf{r}_i$ and feed the new vector to the bi-LSTM model.

In our model, we combine the word embedding, POS embedding and dependency embedding into a single vector to be processed by the bi-LSTM model:

$$\begin{aligned}
\mathbf{x}_i &= \mathbf{w}_i \oplus \mathbf{t}_i \oplus \mathbf{r}_i, \\
\overrightarrow{\mathbf{h}}_i &= \text{LSTM}_{\overrightarrow{\Theta}}(\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{x}_i), \\
\overleftarrow{\mathbf{h}}_i &= \text{LSTM}_{\overleftarrow{\Theta}}(\overleftarrow{\mathbf{h}}_{i+1}, \mathbf{x}_i), \\
\mathbf{h}_i &= \overrightarrow{\mathbf{h}}_i \oplus \overleftarrow{\mathbf{h}}_i,
\end{aligned}$$

where $\oplus$ represents concatenation of vectors, and $\overrightarrow{\Theta}$ and $\overleftarrow{\Theta}$ are parameters of the bi-LSTM model. The complete model is shown in Figure 2.

2.4 Global Inference through ILP

Although the method above has explicitly incorporated some syntactic information into the bi-LSTM model, the syntactic information is used in a soft manner through the learned model weights. We hypothesize that there are also hard constraints we can impose on the compressed sentences. For example, the method above as well as the original method by Filippova et al. (2015) cannot impose any length constraint on the compressed sentences. This is because the labels $y_1, y_2, \ldots, y_n$ are not jointly predicted.

We propose to use Integer Linear Programming (ILP) to find an optimal combination of the labels $y_1, y_2, \ldots, y_n$ for a sentence, subject to some constraints. Specifically, the ILP problem consists of two parts: the objective function, and the constraints.

The Objective Function

Recall that the trained bi-LSTM model above produces a probability distribution for each label y_i, as defined in Eqn. (1). Let us use α_i to denote the probability of $y_i = 1$ as estimated by the bi-LSTM model. Intuitively, we would like to set y_i to 1 if α_i is large.

Besides the probability estimated by the bi-LSTM model, here we also consider the depth of the word w_i in the dependency parse tree of the sentence. Intuitively, a word closer to the root of the tree is more likely to be retained. In order to incorporate this observation, we define $\mathit{dep}(w_i)$ to be the depth of the word w_i in the dependency parse tree of the sentence. The root node of the tree has a depth of 0, an immediate child of the root has a depth of 1, and so on. For example, the dependency parse tree of an example sentence together with the depth of each word is shown in Figure 3. We can see that some of the words that are deleted according to the ground truth have a relatively larger depth, such as the first "she" (with a depth of 4) and the word "flexible" (with a depth of 5).

Based on these considerations, we define the objective function to be the following:

$$\max \quad \sum_{i=1}^{n} y_i(\alpha_i - \lambda \cdot \mathit{dep}(w_i)), \qquad (2)$$

where λ is a positive parameter to be manually set, and y_i is the same as defined before, which is either 0 or 1 to indicate whether w_i is deleted or not.

Constraints

We further introduce some constraints to capture tow considerations. The first consideration is related to the syntactic structure of a sentence, and the second consideration is related to the length of the compressed sentence. Some of the constraints are inspired by Clarke and Lapata (2008).

Our constraints are listed below: (1) No missing parent: Generally, we believe that if a word is retained in the compressed sentence, its parent in the dependency parse tree should also be retained. (2) No missing child: For some dependency relations such as *nsubj*, if the parent word is retained, it makes sense to also keep the child word; otherwise the sentence may become ungrammatical. (3) Max length: Since we are trying to compress a sentence, we may need to impose a minimum compression rate. This could be achieved by setting a maximum value of the sum of y_i. (4) Min length: We observe that the original model sometimes produces very short compressed sentences. We therefore believe that it is also important to maintain a mininum length of the compressed sentence. This can be achieved by setting a minimum value of the sum of y_i.

Formally, the constraints are listed as follows:

$$\begin{array}{rrcl} & \sum_{i=1}^{n} y_i & <= & \beta n, \\ & \sum_{i=1}^{n} y_i & >= & \gamma n, \\ \forall y_i : & y_i & \leq & y_{p_i}, \\ \forall r_i \in \mathcal{T}' : & y_i & \geq & y_{p_i}, \end{array}$$

where w_{p_i} is the parent word of w_i in the dependency parse tree, r_i is the dependency relation type between w_i and w_{p_i}, and $\mathcal{T}'$ is a set of dependency relations for which the child word is often retained when the parent word is retained in the compressed sentence.

The set $\mathcal{T}'$ is derived as follows. For each dependency relation type, based on the training data, we compute the conditional probability of the child word being retained given that the parent word is retained. If this probability is higher than 90%, we include this dependency relation type in $\mathcal{T}'$.

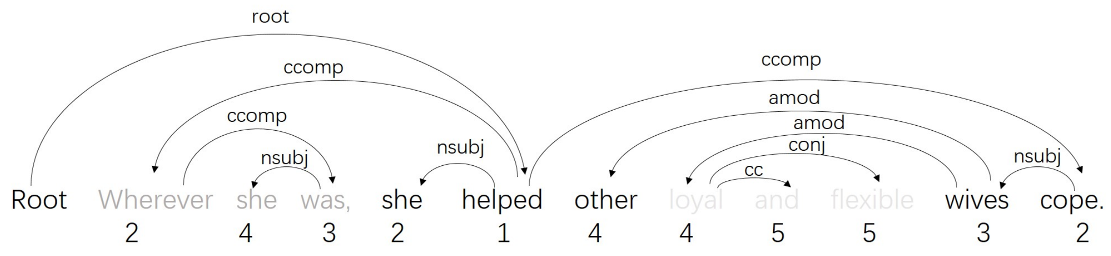

Figure 3: Dependency parse tree of an example sentence. The numbers below the words indicate the depths of the words in the tree. Words in gray are supposed to be deleted based on the ground truth.

3 Experiments

3.1 Datasets and Experiment Settings

Because we are mostly interested in a cross-domain setting where the model is trained on one domain and test on a different domain, we need data from different domains for our evaluation. Here we use three datasets.

***Google News*:** The first dataset contains 10,000 sentence pairs collected and released by Filippova et al. (2015)[2]. The sentences were automatically acquired from the web through Google News using a method introduced by Filippova and Altun (2013). The news articles were from 2013 and 2014.

***BNC News*:** The second dataset contains around 1,500 sentence pairs collected by Clarke and Lapata (2008)[3]. The sentences were from British National Corpus (BNC) and the American News Text corpus before 2008.

***Research Papers*:** The last dataset contains 100 sentences taken from 10 randomly selected papers published at the ACL conference in 2016.

For *Google News* and *BNC News*, we have the ground truth compressed sentences, which are deletion-based compressions, i.e., subsequences of the original sentences. For *Research Papers*, we use it only for manual evaluation in terms of readability and informativeness, as we will explain below.

We evaluate three settings of our method:

BiLSTM: In this setting, we use only the base bi-LSTM model without incorporating any syntactic feature.

BiLSTM+SynFeat: In this setting, we combine word embeddings with POS embeddings and dependency embeddings as input to the bi-LSTM model and use the predictions of $\boldsymbol{y}$ from the bi-LSTM model.

BiLSTM+SynFeat+ILP: In this setting, on top of **BiLSTM+SynFeat**, we solve the ILP problem as described in Section 2.4 to predict the final label sequence $\boldsymbol{y}$.

In the experiments, our model was trained using the Adam (Kingma and Ba, 2015) algorithm with a learning rate initialized at 0.001. The dimension of the hidden layers of bi-LSTM is 100. Word embeddings are initialized from GloVe 100-dimensional pre-trained embeddings (Pennington et al., 2014). POS and dependency embeddings are randomly initialized with 40-dimensional vectors. The embeddings are all updated during training. Dropping probability for dropout layers between stacked LSTM layers is 0.5. The batch size is set as 30. For the ILP part, λ is set to 0.5, β and γ are turned by the validation data and finally they are set to 0.7 and 0.2, respectively. We utilize an open source ILP solver[4] in our method.

We compare our methods with a few baselines:

LSTM: This is the basic LSTM-based deletion method proposed by (Filippova et al., 2015). We report both the performance they achieved using close to two million training sentence pairs and the performance of our re-implementation of their model trained on the 8,000 sentence pairs.

LSTM+: This is advanced version of the model proposed by Filippova et al. (2015), where the authors incorporated some dependency parse tree information into the LSTM model and used the prediction on the previous word to help the prediction on the current word.

Traditional ILP: This is the ILP-based method proposed by Clarke and Lapata (2008). This method does not use neural network models and

[2]Available at `http://storage.googleapis.com/sentencecomp/compression-data.json`.

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

[4]`gnu.org/software/glpk`

is an unsupervised method that relies heavily on the syntactic structures of the input sentences[5].

Abstractive seq2seq: This is an abstractive sequence-to-sequence model trained on 3.8 million Gigaword title-article pairs as described in Section 1.

3.2 Automatic Evaluation

With the two datasets *Google News* and *BNC News* that have the ground truth compressed sentences, we can perform automatic evaluation. We first split the *Google News* dataset into a training set, a validation set and a test set. We took the first 1,000 sentence pairs from *Google News* as the test set, following the same practice as Filippova et al. (2015). We then use 8,000 of the remaining sentence pairs for training and the other 1,000 sentence pairs for validation. For the *NBC News* dataset, we use it only as a test set, applying the sentence compression models trained from the 8,000 sentence pairs from *Google News*.

We use the ground truth compressed sentences to compute accuracy and F1 scores. Accuracy is defined as the percentage of tokens for which the predicted label y_i is correct. F1 scores are derived from precision and recall values, where precision is defined as the percentage of retained words that overlap with the ground truth, and recall is defined as the percentage of words in the ground truth compressed sentences that overlap with the generated compressed sentences.

We report both in-domain performance and cross-domain performance in Table 1. From the table, we have the following observations: (1) For the abstractive sequence-to-sequence model, it was trained on the Gigaword data, so for both *Google News* and *NBC News*, the performance shown is cross-domain performance. We can see that indeed this abstractive method performed poorly in cross-domain settings. (2) In the in-domain setting, with the same amount of training data (8,000), our BiLSTM method with syntactic features (BiLSTM+SynFeat and BiLSTM+SynFeat+ILP) performs similarly to or better than the LSTM+ method proposed by Filippova et al. (2015), in terms of both F1 and accuracy. This shows that our method is comparable to the LSTM+ method in the in-domain setting. (3) In the in-domain setting, even compared with the

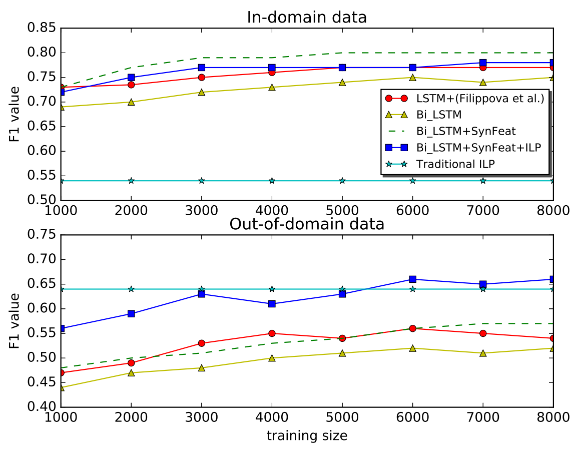

Figure 4: F1 scores with different sizes of training data for in-domain and cross-domain settings.

performance of LSTM+ trained on 2 million sentence pairs, our method trained on 8,000 sentence pairs does not perform substantially worse. (4) In the out-of-domain setting, our BiLSTM+SynFeat and BiLSTM+SynFeat+ILP methods clearly outperform the LSTM and LSTM+ methods. This shows that by incorporating more syntactic features, our methods learn a sentence compression model that is less domain-dependent. (5) The Traditional ILP method also works better than the LSTM and LSTM+ methods in the out-of-domain setting. This is probably because the Traditional ILP method relies heavily on syntax, which is less domain-dependent compared with lexical patterns. But the Traditional ILP method performs worse in the in-domain setting than both the LSTM and LSTM+ methods and our methods.

Overall, Table 1 shows that our proposed method combines both the strength of neural network models in the in-domain setting and the strength of the syntax-based methods in the cross-domain setting. Therefore, our method works reasonably well for both in-domain and out-of-domain data.

We also notice that on *Google News*, adding the ILP layer decreased the sentence compression performance. After some analysis, we think the reason is that some of the constraints used in the ILP layer have led to less deletion but the ground truth compressed sentences in the *Google News* data tend to be shorter compared with those in the *NBC News* data.

We also conduct additional experiments to see the effect of the training data size on our meth-

[5]We use an open source implementation: `https://github.com/cnap/sentence-compression`.

	size of training data	*Google News* F1	*Google News* Acc	*Google News* CR	*NBC News* F1	*NBC News* Acc	*NBC News* CR
LSTM (Filippova et al., 2015)	2 million	0.80	-	0.39	-	-	-
LSTM+ (Filippova et al., 2015)	2 million	**0.82**	-	0.38	-	-	-
Traditional ILP (Clarke and Lapata, 2008)	N/A	0.54	0.56	0.62	0.64	0.56	0.56
Abstractive seq2seq	3.8M	0.09	0.02	0.16	0.14	0.06	0.21
LSTM (our implementation)	8000	0.74	0.75	0.45	0.51	0.48	0.37
LSTM+ (our implementation)	8000	0.77	0.78	0.47	0.54	0.51	0.38
BiLSTM	8000	0.75	0.76	0.43	0.52	0.50	0.34
BiLSTM+SynFeat	8000	0.80	**0.82**	0.43	0.57	0.54	0.37
BiLSTM+SynFeat+ILP	8000	0.78	0.78	0.57	**0.66**	**0.58**	0.53

Table 1: Automatic evaluation of our sentence compression methods. CR standards for compression rate and is defined as the average percentage of words that are retained after compression.

ods and the LSTM+ method. Figure 4 shows the F1 scores on the in-domain *Google News* data and the out-of-domain *NBC News* data when we train the models using different amounts of sentence pairs. We can see that in the in-domain setting, our method does not have any advantage over the LSTM+ method. But in the cross-domain setting, our method that uses ILP to impose syntax-based constraints clearly performs better than LSTM+ when the amount of training data is relatively small.

3.3 Manual Evaluation

The evaluation above does not look at the readability of the compressed sentences. In order to evaluate whether sentences generated by our method are readable, we adopt the manual evaluation procedure by Filippova et al. (2015) to compare our method with LSTM+ and Traditional ILP in terms of readability and informativeness. We asked two raters to score a randomly selected set of 100 sentences from the *Research Papers* dataset. The compressed sentences were randomly ordered and presented to the human raters to avoid any bias. The raters were asked to score the sentences on a five-point scale in terms of both readability and informativeness. We show the average scores of the three methods we compare in Table 3. We can see that our BiLSTM+SynFeat+ILP method clearly outperforms the two baseline methods in the manual evaluation.

We also show a small sample of input sentences from the *Research Papers* dataset and the automatically compressed sentences by different methods in Table 2. As we can see from the table, a general weakness of the LSTM+ method is that the compressed sentences may not be grammatical. In comparison, our method does better in terms of preserving grammaticality.

4 Related Work

Sentence compression can be seen as sentence-level summarization. Similar to document summarization, sentence compression methods can be divided into extractive compression and abstractive compression methods, based on whether words in the compressed sentence all come from the source sentence. In this paper, we focus on deletion-based sentence compression, which is a spacial case of extractive sentence compression.

An early work on sentence compression was done by Jing (2000), who proposed to use several resources to decide whether a phrase in a sentence should be removed or not. Knight and Marcu (2000) proposed to apply a noisy-channel model from machine translation to the sentence compression task, but their model encountered the problem that many SCFG rules have unreliable probability estimates with inadequate data. Galley and McKeown (2007) tried to solve this problem by utilizing parent annotation, Markovization and lexicalization, which have all been shown to improve the quality of the rule probability estimates. Cohn and Lapata (2007) formulated sentence compression as a tree-to-tree rewrite problem. They utilized a synchronous tree substitution grammar (STSG) to license the space of all possible rewrites. Each rule has a weight learned from the training data. For prediction, an algorithm was used to search for the best scoring compression using the gram-

Although dynamic oracles are widely used in dependency parsing and available for most standard transition systems , no dynamic oracle parsing model has yet been proposed for phrase structure grammars
T: Although are used for transition systems model has been proposed for structure grammars .
S: Although dynamic oracles are .
B: Although oracles are used no model has been proposed for structure grammars .

As described above , we used Bayesian Optimization to find optimal hyperparameter configurations in fewer steps than in regular grid search .
T: As described we used Optimization to find configurations in steps in search .
S: As described above Optimization to find optimal hyperparameter configurations steps than in grid search .
B: As described , we used Bayesian Optimization to find optimal hyperparameter configurations in steps.

Following the phrase structure of a source sentence , we encode the sentence recursively in a bottom-up fashion to produce a vector representation of the sentence and decode it while aligning the input phrases and words with the output .
T: Following structure of a sentence we encode sentence recursively to produce a representation of the sentence and decode it while aligning phrases and words with output .
S: Following the structure of a source sentence encode the sentence recursively in a bottom-up fashion .
B: Following the structure , we encode the sentence recursively in a bottom-up fashion to produce a vector representation and decode it .

Table 2: Some input sentences from the *Research Papers* dataset and the automatically compressed sentences using different methods. T: Traditional ILP method. S: LSTM+. B: BiLSTM+SynFeat+ILP.

	readability	informativeness
Traditional ILP	3.94	3.33
LSTM+	3.69	3.07
BiLSTM+SynFeat+ILP	4.29	3.46

Table 3: Manual evaluation.

mar rules. Besides, Cohn and Lapata (2008) extended this model to abstractive sentence compression, which includes substitution, reordering and insertion. McDonald (2006) proposed a graph-based sentence compression method. The general idea is that each word pair in the original sentence has a score. The task then becomes how to find a compressed sentence with a length limit according word pair scores. Their method is similar to graph-based dependency parsing. Clarke and Lapata (2008) first used an ILP framework for sentence compression. In the paper, the author put forward three models. The first model is a language model reformulated by ILP. As the first model treats all the words equally, the second model uses a corpus to learn an importance score for each word and then incorporates it in the ILP model. The Last model, which is based on (McDonald, 2006), replaces the decoder with an ILP model and adds many linguistic constraints such as dependency parsing compared with the previous two ILP models. Filippova and Strube (2008) represented sentences with dependency parse trees and an ILP-based method was used to decide whether the dependencies were preserved or not. Different from most previous work that treats sentence extraction and sentence compression separately, Berg-Kirkpatrick et al. (2011) jointly model the two processes in one ILP problem. Bigrams and subtrees are represented by some features, and feature are learned on some training data. The ILP problem maximizes the coverage of weighted bigrams and deleted subtrees of the summary.

In recent years, neural network models, especially sequence-to-sequence models, have been applied to sentence compression. Our work is based on the deletion-based LSTM model for sentence compression by Filippova et al. (2015). There has also been much interest in applying sequence-to-sequence models for abstractive sentence compression (Rush et al., 2015; Chopra et al., 2016). As we pointed out in Section 1, in a cross-domain setting, abstractive sentence compression may not be suitable.

5 Conclusions

In this paper, we studied how to modify an LSTM model for deletion-based sentence compression so that the model works well in a cross-domain setting. We hypothesized that incorporation of syntactic information into the training of the LSTM model would help. We thus proposed two ways to incorporate syntactic information, one through directly adding POS tag embeddings and dependency type embeddings, and the other through the objective function and constraints of an Integer Linear Programming (ILP) model. The experiments showed that our proposed bi-LSTM model with syntactic features and an ILP layer works well in both in-domain and cross-domain settings. In comparison, the original LSTM model does not work well in the cross-domain setting, and a traditional ILP method does not work well in the in-domain setting. Therefore, our proposed method is relatively more robust than these baselines. We also manually evaluated the compressed sentences generated by our method and found that the method works better than the baselines in terms of both readability and informativeness.

Acknowledgment

This work is supported by DSO grant DSOCL15223. The work was conducted during the first author's visit to the Singapore Management University.

References

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research* .

Trevor Cohn and Mirella Lapata. 2007. Large margin synchronous generation and its application to sentence compression. In *EMNLP-CoNLL*. pages 73–82.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*.

Michel Galley and Kathleen McKeown. 2007. Lexicalized markov grammars for sentence compression. In *HLT-NAACL*. pages 180–187.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the sixth conference on Applied natural language processing*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization step one: Sentence compression. In *Proceedings of the 17th National Conference on Artificial Intelligence*.

Ryan T McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *European Chapter of the Association for Computational Linguistics Valencia*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.