Sentence Compression by Deletion with LSTMs

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, Oriol Vinyals
Google Research
{katjaf,ealfonseca,crcarlos,lukaszkaiser,vinyals}@google.com

Abstract

We present an LSTM approach to deletion-based sentence compression where the task is to translate a sentence into a sequence of zeros and ones, corresponding to token deletion decisions. We demonstrate that even the most basic version of the system, which is given no syntactic information (no PoS or NE tags, or dependencies) or desired compression length, performs surprisingly well: around 30% of the compressions from a large test set could be regenerated. We compare the LSTM system with a competitive baseline which is trained on the same amount of data but is additionally provided with all kinds of linguistic features. In an experiment with human raters the LSTM-based model outperforms the baseline achieving 4.5 in readability and 3.8 in informativeness.

1 Introduction

Sentence compression is a standard NLP task where the goal is to generate a shorter paraphrase of a sentence. Dozens of systems have been introduced in the past two decades and most of them are deletion-based: generated compressions are token subsequences of the input sentences (Jing, 2000; Knight & Marcu, 2000; McDonald, 2006; Clarke & Lapata, 2008; Berg-Kirkpatrick et al., 2011, to name a few).

Existing compression systems heavily use syntactic information to minimize chances of introducing grammatical mistakes in the output. A common approach is to use only some syntactic information (Jing, 2000; Clarke & Lapata, 2008, among others) or use syntactic features as signals in a statistical model (McDonald, 2006). It is probably even more common to operate on syntactic trees directly (dependency or constituency) and generate compressions by pruning them (Knight & Marcu, 2000; Berg-Kirkpatrick et al., 2011; Filippova & Altun, 2013, among others). Unfortunately, this makes such systems vulnerable to error propagation as there is no way to recover from an incorrect parse tree. With the state-of-the-art parsing systems achieving about 91 points in labeled attachment accuracy (Zhang & McDonald, 2014), the problem is not a negligible one. To our knowledge, there is no competitive compression system so far which does not require any linguistic preprocessing but tokenization.

In this paper we research the following question: can a robust compression model be built which only uses tokens and has no access to syntactic or other linguistic information? While phenomena like long-distance relations may seem to make generation of grammatically correct compressions impossible, we are going to present an evidence to the contrary. In particular, we will present a model which benefits from the very recent advances in deep learning and uses word embeddings and Long Short Term Memory models (LSTMs) to output surprisingly readable and informative compressions. Trained on a corpus of less than two million automatically extracted parallel sentences and using a standard tool to obtain word embeddings, in its best and most simple configuration it achieves 4.5 points out of 5 in readability and 3.8 points in informativeness in an extensive evaluation with human judges. We believe that this is an important result as it may suggest a new direction for sentence compression research which is less tied to modeling linguistic

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 360–368,
Lisbon, Portugal, 17-21 September 2015.

structures, especially syntactic ones, than the compression work so far.

The paper is organized as follows: Section 3 presents a competitive baseline which implements the system of McDonald (2006) for large training sets. The LSTM model and its three configurations are introduced in Section 4. The evaluation set-up and a discussion on wins and losses with examples are presented in Section 5 which is followed by the conclusions.

2 Related Work

The problem formulation we adopt in this paper is very simple: for every token in the input sentence we ask whether it should be kept or dropped, which translates into a sequence labeling problem with just two labels: one and zero. The deletion approach is a standard one in compression research, although the problem is often formulated over the syntactic structure and not the raw token sequence. That is, one usually drops constituents or prunes dependency edges (Jing, 2000; Knight & Marcu, 2000; McDonald, 2006; Clarke & Lapata, 2008; Berg-Kirkpatrick et al., 2011; Filippova & Altun, 2013). Thus, the relation to existing compression work is that we also use the deletion approach.

Recent advances in machine learning made it possible to escape the typical paradigm of mapping a fixed dimensional input to a fixed dimensional output to mapping an input sequence onto an output sequence. Even though many of these models were proposed more than a decade ago, it is not until recently that they have empirically been shown to perform well. Indeed, core problems in natural language processing such as translation (Cho et al., 2014; Sutskever et al., 2014; Luong et al., 2014), parsing (Vinyals et al., 2014), image captioning (Vinyals et al., 2015; Xu et al., 2015), or learning to execute small programs (Zaremba & Sutskever, 2014) employed virtually the same principles—the use of Recurrent Neural Networks (RNNs). Thus, with regard to this line of research, our work comes closest to the recent machine translation work. An important difference is that we do not aim at building a model that generates compressions directly but rather a model which generates a sequence of deletion decisions.

A more complex translation model is also conceivable and may significantly advance work on compression by paraphrasing, of which there have not been many examples yet (Cohn & Lapata, 2008). However, in this paper our goal is to demonstrate that a simple but robust deletion-based system can be built without using any linguistic features other than token boundaries. We leave experiments with paraphrasing models to future work.

3 Baseline

We compare our model against the system of McDonald (2006) which also formulates sentence compression as a binary sequence labeling problem. In contrast to our proposal, it makes use of a large set of syntactic features which are treated as soft evidence. The presence or absence of these features is treated as signals which do not condition the output that the model can produce. Therefore the model is robust against noise present in the precomputed syntactic structures of the input sentences.

The system was implemented based on the description by McDonald (2006) with two changes which were necessary due to the large size of the training data set used for model fitting. The first change was related to the learning procedure and the second one to the family of features used.

Regarding the learning procedure, the original model uses a large-margin learning framework, namely MIRA (Crammer & Singer, 2003), but with some minor changes as presented by McDonald et al. (2005). In this set-up, online learning is performed, and at each step an optimization procedure is made where K constraints are included, which correspond to the top-K solutions for a given training observation. This optimization step is equivalent to a Quadratic Programming problem if $K > 1$, which is time-costly to solve, and therefore not adequate for the large amount of data we used for training the model. Furthermore, in his publication McDonald states clearly that different values of K did not actually have a major impact on the final performance of the model. Consequently, and for the sake of being able to successfully train the model with large-scale data, the learning procedure is implemented as a distributed structured perceptron with iterative parameter mixing (McDonald et al., 2010), where each shard is processed with MIRA and K is set to 1.

Setting $K = 1$ will only affect the weight update described on line 4 of Figure 3 of McDonald

(2006), which is now expressed as:

$$\boldsymbol{w}^{(i+1)} \leftarrow \boldsymbol{w}^{(i)} + \tau \times \boldsymbol{e}_{\boldsymbol{y}_t,\boldsymbol{y}'}$$
$$\text{where } \tau = \max\left(0, \frac{L(\boldsymbol{y}_t, \boldsymbol{y}') - \boldsymbol{w} \cdot \boldsymbol{e}_{\boldsymbol{y}_t,\boldsymbol{y}'}}{||\boldsymbol{e}_{\boldsymbol{y}_t,\boldsymbol{y}'}||^2}\right)$$
$$\boldsymbol{e}_{\boldsymbol{y}_t,\boldsymbol{y}'} = \boldsymbol{F}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{F}(\boldsymbol{x}_t, \boldsymbol{y}')$$
$$\boldsymbol{y}' = \text{best}(\boldsymbol{x}; \boldsymbol{w}^{(i)})$$
$$\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=2}^{|\boldsymbol{y}|} \boldsymbol{f}(\boldsymbol{x}, I(y_{j-1}), I(y_j))$$

The second change concerns the feature set used. While McDonald's original model contains deep syntactic features coming from both dependency and constituency parse trees, we use only dependency-based features. Additionally, and to better compare the baseline with the LSTM models, we have included as an optional feature a 256-dimension embedding-vector representation of each input word and its syntactic parent. The vectors are pre-trained using the Skipgram model[1] (Mikolov et al., 2013). Ultimately, our implementation of McDonald's model contained 463,614 individual features, summarized in three categories:

- **PoS features:** Joint PoS tags of selected tokens. Unigram, bigram and trigram PoS context of selected and dropped tokens. All the previous features conjoined with one indicating if the last two selected tokens are adjacent.
- **Deep syntactic features:** Dependency labels of taken and dropped tokens and their parent dependencies. Boolean features indicating syntactic relations between selected tokens (*i.e.*, siblings, parents, leaves, etc.). Dependency label of the least common ancestor in the dependency tree between a batch of dropped tokens. All the previous features conjoined with the PoS tag of the involved tokens.
- **Word features:** Boolean features indicating if a group of dropped nodes contain a complete or incomplete parenthesization. Word-embedding vectors of selected and dropped tokens and their syntactic parents.

The model is fitted over ten epochs on the whole training data, and for model selection a small development set consisting of 5,000 previously unseen sentences is used (none of them belonging to the evaluation set). The automated metric used for this selection was accuracy@1 which is the proportion of golden compressions which could be fully reproduced. The performance on the development set plateaus when getting close to the last epoch.

[1] https://code.google.com/p/word2vec/

4 The LSTM model

Our approach is largely based on the sequence to sequence paradigm proposed in Sutskever et al. (2014). We train a model that maximizes the probability of the correct output given the input sentence. Concretely, for each training pair (X, Y), we will learn a parametric model (with parameters θ), by solving the following optimization problem:

$$\theta^* = \arg\max_{\theta} \sum_{X,Y} \log p(Y|X;\theta) \quad (1)$$

where the sum is assumed to be over all training examples. To model the probability p, we use the same architecture described by Sutskever et al. (2014). In particular, we use a RNN based on the Long Short Term Memory (LSTM) unit (Hochreiter & Schmidhuber, 1997), designed to avoid vanishing gradients and to remember some long-distance dependences from the input sequence. Figure 1 shows a basic LSTM architecture. The RNN is fed with input words X_i (one at a time), until we feed a special symbol "GO". It is now a common practice (Sutskever et al., 2014; Li & Jurafsky, 2015) to start feeding the input in reversed order, as it has been shown to perform better empirically. During the first pass over the input, the network is expected to learn a compact, distributed representation of the input sentence, which will allow it to start generating the right predictions when the second pass starts, after the "GO" symbol is read.

We can apply the chain rule to decompose Equation (1) as follows:

$$p(Y|X;\theta) = \prod_{t=1}^{T} p(Y_t|Y_1, \ldots, Y_{t-1}, X; \theta) \quad (2)$$

noting that we made no independence assumptions. Once we find the optimal θ^*, we construct our estimated compression $\hat{Y}$ as:

$$\hat{Y} = \arg\max_{Y} p(Y|X;\theta^*) \quad (3)$$

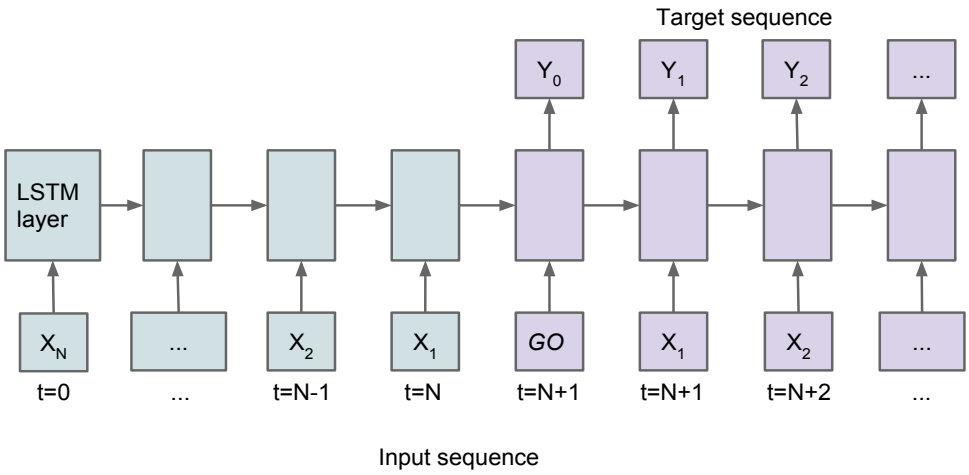

Figure 1: High-level overview of an LSTM unrolled through time.

LSTM cell: Let us review the sequence-to-sequence LSTM model. The Long Short Term Memory model of Hochreiter & Schmidhuber (1997) is defined as follows. Let x_t, h_t, and m_t be the input, control state, and memory state at timestep t. Then, given a sequence of inputs $(x_1, \ldots, x_T)$, the LSTM computes the h-sequence $(h_1, \ldots, h_T)$ and the m-sequence $(m_1, \ldots, m_T)$ as follows

$$\begin{aligned} i_t &= \text{sigm}(W_1 x_t + W_2 h_{t-1}) \\ i'_t &= \tanh(W_3 x_t + W_4 h_{t-1}) \\ f_t &= \text{sigm}(W_5 x_t + W_6 h_{t-1}) \\ o_t &= \text{sigm}(W_7 x_t + W_8 h_{t-1}) \\ m_t &= m_{t-1} \odot f_t + i_t \odot i'_t \\ h_t &= m_t \odot o_t \end{aligned}$$

The operator $\odot$ denotes element-wise multiplication, the matrices $W_1, \ldots, W_8$ and the vector h_0 are the parameters of the model, and all the non-linearities are computed element-wise.

Stochastic gradient descent is used to maximize the training objective (Eq. (1)) w.r.t. all the LSTM parameters.

Network architecture: In these experiments we have used the architecture depicted in Figure 3. Following Vinyals et al. (2014), we have used three stacked LSTM layers to allow the upper layers to learn higher-order representations of the input, interleaved with dropout layers to prevent overfitting (Srivastava et al., 2014). The output layer is a SoftMax classifier that predicts, after the "GO" symbol is read, one of the following three

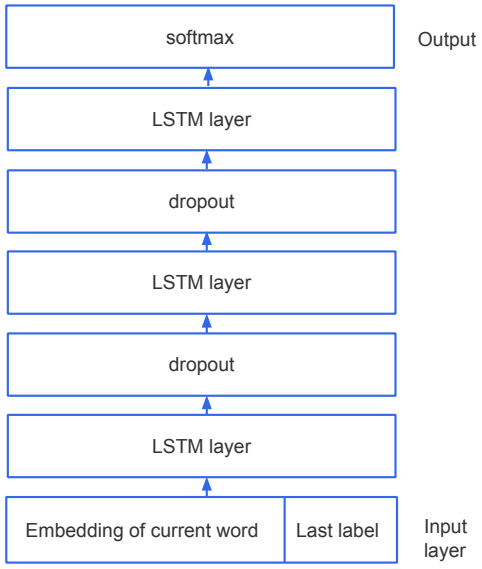

Figure 3: Architecture of the network used for sentence compression. Note that this basic structure is then unrolled 120 times, with the standard dependences from LSTM networks (Hochreiter & Schmidhuber, 1997).

labels: **1**, if a word is to be retained in the compression, **0** if a word is to be deleted, or **EOS**, which is the output label used for the "GO" input and the end-of-sentence final period.

Input representation: In the simplest implementation, that we call LSTM, the input layer has 259 dimensions. The first 256 contain the embedding-vector representation of the current in-

```
function DECODE(X)
                                          ▷ Initialize and feed the reversed input.
    Lstm ← CREATELSTM
    LayersState ← INITIALIZELAYERS(Lstm)
    for all X_i ∈ REVERSE(X) do
        LayersState ← ONESTEP(Lstm, LayersState, X_i)
    end for
    LayersState ← ONESTEP(Lstm, LayersState, GO)
      ▷ Create the beam vector. Each item contains the state of the layers, the labels predicted so far, and probability.
    Beam ← {(LayersState, (), 1.0)}
                                                          ▷ Beam search
    for all X_i ∈ X do
        NextBeam ← {}
        for all (LayersState, Labels, Prob) ∈ Beam do
            (NextLayersState, Outputs) ← ONESTEP(Lstm, LayersState, X_i)
            for all Output ∈ Outputs do
                NextBeam ← NextBeam∪{(NextLayerState, Labels+Output.label, Prob∗Output.prob)}
            end for
        end for
        Beam ← TOPN(NextBeam)
    end for
    return TOP(Beam)
end function
```

Figure 2: Pseudocode of the beam-search algorithm for compressing an input sentence.

put word, pre-trained using the Skipgram model[2] (Mikolov et al., 2013). The final three dimensions contain a one-hot-spot representation of the gold-standard label of the previous word (during training), or the generated label of the previous word (during decoding).

For the LSTM+PAR architecture we first parse the input sentence, and then we provide as input, for each input word, the embedding-vector representation of that word and its parent word in the dependency tree. If the current input is the root node, then a special parent embedding is constructed with all nodes set to zero except for one node. In these settings we want to test the hypothesis whether knowledge about the parent node can be useful to decide if the current constituent is relevant or not for the compression. The dimensionality of the input layer in this case is 515. Similarly to McDonald (2006), syntax is used here as a soft feature in the model.

For the LSTM+PAR+PRES architecture, we again parse the input sentence, and use a 518-sized embedding vector, that includes:

- The embedding vector for the current word (256 dimensions).
- The embedding vector for the parent word (256 dimensions).
- The label predicted for the last word (3 dimensions).
- A bit indicating whether the parent word has already been seen and kept in the compression (1 dimension).
- A bit indicating whether the parent word has already been seen but discarded (1 dimension).
- A bit indicating whether the parent word comes later in the input (1 dimension).

Decoding: Eq. (3) involves searching through all possible output sequences (given X). Contrary to the baseline, in the case of LSTMs the complete previous history is taken into account for each prediction and we cannot simplify Eq. (2) with a Markov assumption. Therefore, the search space at decoding time is exponential on the length of the input, and we have used a beam-search procedure as described in Figure 2.

Fixed parameters: For training, we unfold the network 120 times and make sure that none of our training instances is longer than that. The learning rate is initialized at 2, with a decay factor of 0.96 every 300,000 traning steps. The dropping probability for the dropout layers is 0.2. The number of nodes in each LSTM layer is always identical to the number of nodes in the input layer. We have not tuned these parameters nor the number of stacked layers.

[2] https://code.google.com/p/word2vec/

5 Evaluation

5.1 Data

Both the LSTM systems we introduced and the baseline require a training set of a considerable size. In particular, the LSTM model uses 256-dimensional embeddings of token sequences and cannot be expected to perform well if trained on a thousand parallel sentences, which is the size of the commonly used data sets (Knight & Marcu, 2000; Clarke & Lapata, 2006). Following the method of Filippova & Altun (2013), we collect a much larger corpus of about two million parallel sentence-compression instances from the news where every compression is a subsequence of tokens from the input. For testing, we use the publicly released set of 10,000 sentence-compression pairs[3]. We take the first 200 sentences from this set for the manual evaluation with human raters, and the first 1,000 sentences for the automatic evaluation.

5.2 Experiments

We evaluate the baseline and our systems on the 200-sentence test set in an experiment with human raters. The raters were asked to rate *readability* and *informativeness* of compressions given the input which are the standard evaluation metrics for compression. The former covers the grammatical correctness, comprehensibility and fluency of the output while the latter measures the amount of important content preserved in the compression.

Additionally, for experiments on the development set, we used two metrics for automatic evaluation: *per-sentence accuracy* (i.e., how many compressions could be fully reproduced) and *word-based F1-score*. The latter differs from the RASP-based relation F-score by Riezler et al. (2003) in that we simply compute the recall and precision in terms of tokens kept in the golden and the generated compressions. We report these results for completeness although it is the results of the human evaluation from which we draw our conclusions.

Compression ratio: The three versions of our system (LSTM*) and the baseline (MIRA) have comparable compression ratios (CR) which are defined as the length of the compression in characters divided over the sentence length. Since the ratios are very close, a comparison of the systems' scores is justified (Napoles et al., 2011).

Automatic evaluation: A total of 1,000 sentence pairs from the test set[4] were used in the automatic evaluation. The results are summarized in Table 1.

	F1	*Acc*	*CR*
MIRA	0.75	0.21	0.37
LSTM	0.80	0.30	0.39
LSTM+PAR	0.81	0.31	0.38
LSTM+PAR+PRES	0.82	0.34	0.38

Table 1: F1-score, per-sentence accuracy and compression ratio for the baseline and the systems

There is a significant difference in performance of the MIRA baseline and the LSTM models, both in terms of F1-score and in accuracy. More than 30% of golden compressions could be fully regenerated by the LSTM systems which is in sharp contrast with the 20% of MIRA. The differences in F-score between the three versions of LSTM are not significant, all scores are close to 0.81.

Evaluation with humans: The first 200 sentences from the set of 1,000 used in the automatic evaluation were compressed by each of the four systems. Every sentence-compression pair was rated by three raters who were asked to select a rating on a five-point Likert scale, ranging from one to five. In very few cases (around 1%) the ratings were inconclusive (i.e., *1, 3, 5* were given to the same pair) and had to be skipped. Table 2 summarizes the results.

	read	*info*
MIRA	4.31	3.55
LSTM	4.51†	3.78†
LSTM+PAR	4.40	3.73
LSTM+PAR+PRES	4.37	3.79†

Table 2: Readability and informativeness for the baseline and the systems: † stands for *significantly better than MIRA with 0.95 confidence*.

The results indicate that the LSTM models produce more readable and more informative compressions. Interestingly, there is no benefit in using the syntactic information, at least not with

[3] http://storage.googleapis.com/sentencecomp/compressiondata.json

[4] We used the very first 1,000 instances.

Sentence & LSTM *Compression*	difficulty
A Virginia state senator and one-time candidate for governor stabbed by his son said Friday that he is "alive so must live," his first public statement since the assault and his son's suicide shortly thereafter. *State senator alive so must live*.	quotes
Gwyneth Paltrow, 41 and husband Chris Martin, 37 are to separate after more than 10 years of marriage, the actress announced on her website GOOP. *Gwyneth Paltrow are to separate*.	commas
Chris Hemsworth and the crew of his new movie 'In the Heart of the Sea' were forced to flee flash floods in the Canary Islands yesterday. *Chris Hemsworth were forced to flee flash floods*.	quotes
Police in Deltona, Fla., are trying to sniff out the identity of a man who allegedly attempted to pay his water bill with cocaine. *Police are trying to sniff out the identity.*	nothing to remove
Just a week after a CISF trooper foiled a suicide bid by a woman in the Delhi metro, another woman trooper from the same force prevented two women commuters from ending their lives, an official said Monday. *Another woman trooper prevented two women commuters.*	important context
Whatever the crisis or embarrassment to his administration, Pres. Obama don't know nuttin' about it. *Pres. Obama don't know nuttin.*	nothing to remove
TRADE and Industry Minister Rob Davies defended the government's economic record in Parliament on Tuesday, saying it had implemented structural reforms and countercyclical infrastructure projects to help shore up the economy. *Rob Davies defended the government's economic record.*	
Social activist Medha Patkar on Monday extended her "complete" support to Arvind Kejriwal-led Aam Aadmi Party in Maharashtra. *Medha Patkar extended her support to Aam Aadmi Party.*	
State Sen. Stewart Greenleaf discusses his proposed human trafficking bill at Calvery Baptist Church in Willow Grove Thursday night. *Stewart Greenleaf discusses his human trafficking bill.*	
Alan Turing, known as the father of computer science, the codebreaker that helped win World War 2, and the man tortured by the state for being gay, is to receive a pardon nearly 60 years after his death. *Alan Turing is to receive a pardon.*	
Robert Levinson, an American who disappeared in Iran in 2007, was in the country working for the CIA, according to a report from the Associated Press's Matt Apuzzo and Adam Goldman. *Robert Levinson was working for the CIA.*	

Figure 4: Example sentences and compressions.

the amount of parallel data we had at our disposal. The simple LSTM model which only uses token embeddings to generate a sequence of deletion decisions significantly outperforms the baseline which was given not only embeddings but also syntactic and other features.

Discussion: What are the wins and losses of the LSTM systems? Figure 4 presents some of the evaluated sentence-compression pairs. In terms of readability, the basic LSTM system performed surprisingly well. Only in a few cases (out of 200) did it get an average score of two or three. Sentences which pose difficulty to the model are the ones with quotes, intervening commas, or other uncommon punctuation patterns. For example, in the second sentence in Figure 4, if one removes from the input the age modifiers and the preceding commas, the words *and Chris Martin* are not dropped and the output compression is grammatical, preserving both conjoined elements.

With regard to informativeness, the difficult cases are those where there is very little to be removed and where the model still removed more than a half to achieve the compression ratio it observed in the training data. For example, the only part that can be removed from the fourth sentence in Figure 4 is the modifier of *police*, everything else being important content. Similarly, in the fifth sentence the context of the event must be retained in the compression for the event to be interpreted correctly.

Arguably, such cases would also be difficult for other systems. In particular, recognizing when the context is crucial is a problem that can be solved only by including deep semantic and discourse features which has not been attempted yet. And

sentences with quotes (direct speech, a song or a book title, etc.) are challenging for parsers which in turn provide important signals for most compression systems.

The bottom of Figure 4 contains examples of good compressions. Even though for a significant number of input sentences the compression was a continuous subsequence of tokens, there are many discontinuous compressions. In particular, the LSTM model learned to drop appositions, no matter how long they are, temporal expressions, optional modifiers, introductory clauses, etc.

Our understanding of why the extended model (LSTM+PAR+PRES) performed worse in the human evlauation than the base model is that, in the absence of syntactic features, the basic LSTM learned a model of syntax useful for compression, while LSTM++, which was given syntactic information, learned to optimize for the particular way the "golden" set was created (tree pruning). While the automatic evaluation penalized all deviations from the single golden variant, in human evals there was no penalty for readable alternatives.

6 Conclusions

We presented, to our knowledge, a first attempt at building a competitive compression system which is given no linguistic features from the input. The two important components of the system are (1) word embeddings, which can be obtained by anyone either pre-trained, or by running `word2vec` on a large corpus, and (2) an LSTM model which draws on the very recent advances in research on RNNs. The training data of about two million sentence-compression pairs was collected automatically from the Internet.

Our results clearly indicate that a compression model which is not given syntactic information explicitly in the form of features may still achieve competitive performance. The high readability and informativeness scores assigned by human raters support this claim. In the future, we are planning to experiment with more "interesting" paraphrasing models which translate the input not into a zero-one sequence but into words.

References

Berg-Kirkpatrick, T., D. Gillick & D. Klein (2011). Jointly learning to extract and compress. In *Proc. of ACL-11*.

Cho, K., B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk & Y. Bengio (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014.

Clarke, J. & M. Lapata (2006). Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proc. of COLING-ACL-06*, pp. 377–385.

Clarke, J. & M. Lapata (2008). Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Cohn, T. & M. Lapata (2008). Sentence compression beyond word deletion. In *Proc. of COLING-08*, pp. 137–144.

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

Filippova, K. & Y. Altun (2013). Overcoming the lack of parallel data in sentence compression. In *Proc. of EMNLP-13*, pp. 1481–1491.

Hochreiter, S. & J. Schmidhuber (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jing, H. (2000). Sentence reduction for automatic text summarization. In *Proc. of ANLP-00*, pp. 310–315.

Knight, K. & D. Marcu (2000). Statistics-based summarization – step one: Sentence compression. In *Proc. of AAAI-00*, pp. 703–711.

Li, J. & D. Jurafsky (2015). A hierarchical LSTM autoencoder for paragraphs and documents. In *Proc. of ACL-15*.

Luong, M.-T., I. Sutskever, Q. V. Le, O. Vinyals & W. Zaremba (2014). Addressing the rare word problem in neural machine translation. *CoRR*, abs/1410.8206.

McDonald, R. (2006). Discriminative sentence compression with soft syntactic evidence. In *Proc. of EACL-06*, pp. 297–304.

McDonald, R., K. Crammer & F. Pereira (2005). Online large-margin training of dependency parsers. In *Proc. of ACL-05*, pp. 91–98.

McDonald, R., K. Hall & G. Mann (2010). Distributed training strategies for the structured perceptron. In *Proc. of NAACL-HLT-10*, pp. 456–464.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado & J. Dean (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pp. 3111–3119.

Napoles, C., C. Callison-Burch & B. Van Durme (2011). Evaluating sentence compression: Pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-to-text Generation,* Portland, OR, June 24 2011, pp. 91–97.

Riezler, S., T. H. King, R. Crouch & A. Zaenen (2003). Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for Lexical-Functional Grammar. In *Proc. of HLT-NAACL-03*, pp. 118–125.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever & R. Salakhutdinov (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Sutskever, I., O. Vinyals & Q. V. Le (2014). Sequence to sequence learning with neural networks. In *Proc. of NIPS-2014*.

Vinyals, O., A. Toshev, S. Bengio & D. Erhan (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR-2015*.

Vinyals, O., L. Kaiser, T. Koo, S. Petrov, I. Sutskever & G. E. Hinton (2014). Grammar as a foreign language. *CoRR*, abs/1412.7449.

Xu, K., J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel & Y. Bengio (2015). Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of ICML-2015*.

Zaremba, W. & I. Sutskever (2014). Learning to execute. *CoRR*, abs/1410.4615.

Zhang, H. & R. McDonald (2014). Enforcing structural diversity in cube-pruned dependency parsing. In *Proc. of ACL-14*, pp. 656–661.