

Sentence Compression as a Supervised Learning with a Rich Feature Space

Elena Churkin and Mark Last and Marina Litvak and Natalia Vanetik

Department of Software Engineering, Shamoon Academic College, Beer Sheva, Israel
{marinal, natalyav}@sce.ac.il, elena.chk@gmail.com

Department of Software and Information Systems Engineering, Ben Gurion University of the
Negev, Beer Sheva, Israel
mlast@bgu.ac.il

Abstract. We present a novel supervised approach to sentence compression, based on classification and removal of word sequences generated from subtrees of the original sentence dependency tree. Our system may use any known classifier like Support Vector Machines or Logistic Model Tree to identify word sequences that can be removed without compromising the grammatical correctness of the compressed sentence. We trained our system using several classifiers on a small annotated dataset of 100 sentences, which included around 1500 manually labeled subtrees (removal candidates) represented by 25 features. The highest cross-validation classification accuracy of 80% was obtained with the SMO (Normalized Poly Kernel) algorithm. We evaluated the readability and the informativeness of the sentences compressed by the SMO-based classification model with the help of human raters using a separate benchmark dataset of 200 sentences.

1 Introduction

Sentence compression is a common NLP task of shortening sentences without changing their meaning and while preserving their correct grammatical structure. One of the important applications of sentence compression is automatic text summarization [11,14]. The other known applications are headline generation [5], generation of television subtitles [18], automatic tweet generation [17], and displaying texts on small screens like mobile phones [6]. Most approaches to the sentence compression task are based on removing words from the sentences, but systems that use paraphrasing also exist [4]. Common sentence compression methods include integer linear programming [2], noisy-channel models [13,10], models that use pruning of dependency and constituency parse trees [12,1] and discriminative large margin learning [16,3]. One of the latest sentence compression methods is based on a probabilistic model (LSTM - Long Short Term Memory) and it does not utilize any syntactic information (like PoS or parse trees) nor the desired compression length in order to compress sentences [7]. However, it builds upon a large set of features and thus requires a training corpus of a considerable size. In [7], the LSTM system is trained on two million sentence-compression instances.

In this paper, we present a new supervised sentence compression method based on detection, labeling, and removing appropriate word sequences (subtrees) from the original sentence. Similarly to the cited works, our approach uses dependency structure of a

sentence and prunes word sequences corresponding to dependency subtrees. However, our approach is totally supervised and does not contain any ad-hoc rules. The subtrees of the dependency parse tree are treated as removal candidates. For each subtree, we calculate 25 predictive features that are later filtered by a feature selection method. Our system may use one of the known classification methods (like SVM) in order to identify subtrees that can be removed without compromising the grammatical correctness of the compressed sentence. Due to a limited amount of predictive features, our system may be trained on a much smaller set of compressed sentences than LSTM-based algorithms. Moreover, the system compression ratio may be controlled by the user. We trained our system with several classifiers on a manually labeled dataset of 1494 subtrees extracted from 100 sentences and evaluated it on a separate test set of 200 sentences. Using dependency tree pruning for sentence compression is motivated by the belief that the grammatical correctness of the compressed sentences can be better ensured by pruning of dependency trees, because tree pruning approaches do not generate new dependencies and are unlikely to produce a compression with a totally different meaning [8]. An unsupervised dependency tree pruning approach is used in [8], whereas in this paper we propose a new supervised tree pruning approach for sentence compression. The current version of our system has been trained on English texts but in principle, it can be trained on an annotated collection of parsed sentences in any language.

2 The Compression Methodology

Our sentence compression methodology is composed of several stages: (1) extraction of removal candidates (word sequences represented by subtrees of a dependency parsing tree), (2) representation of extracted candidates by predictive features (including feature extraction and selection), and (3) candidates classification (valid/invalid for removal) using a pre-trained classification model. Figure 1 demonstrates the general pipeline of our compression procedure.

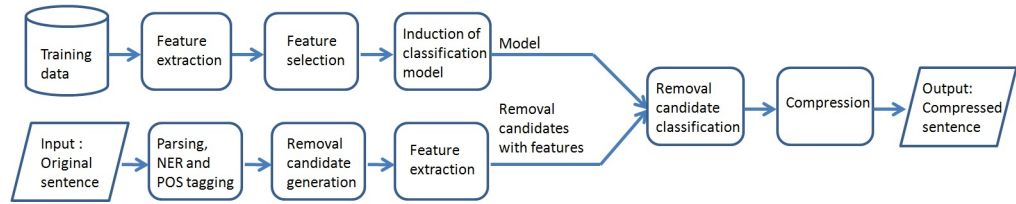


Fig. 1. Pipeline of our sentence compression tool.

2.1 Generation of subtrees (removal candidates)

For each sentence, our system identifies as removal candidates all word sequences representing the possible subtrees of the dependency syntax tree.¹ Also, subtrees' complements and single words (remaining after filtering) are added to the list of candidates. The dependency syntax trees are generated using the Stanford Parser from the Stanford coreNLP tool [15].

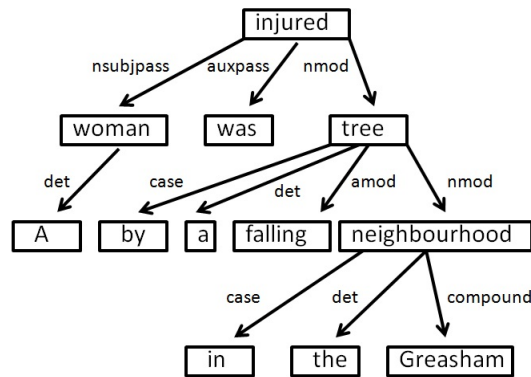


Fig. 2. Dependency syntax tree representing the sentence "A woman was injured by a falling tree in the Gresham neighborhood".

An example of a dependency syntax tree for the sentence "A woman was injured by a falling tree in the Gresham neighborhood" is demonstrated in Figure 2 and selected removal candidates are shown in Table 1, along with the compressed sentences after removal of the corresponding subtree, its type (single word, subtree, or its complement) and its class label ("Y" - valid, "N" - invalid). The sentence has 11 subtrees and 12 single words. Determiners, prepositions, and "be" verbs are filtered, retaining the set of only five removal candidates and their complements for annotation or classification.

Removal candidate	Compressed sentence	Type	Class
falling	A woman was injured by a tree in the Gresham neighborhood.	word	Y
by a falling tree in the Gresham neighborhood	A woman was injured.	subtree	Y
A woman was injured	by a falling tree in the Gresham neighborhood.	complement	N

Table 1. Some possible candidates with their class labels (manually annotated).

¹ Complete sentences and single words that are prepositions, wh-words, pronouns, and forms of the "to be" verb are not saved in the list of removal candidates.

2.2 Feature extraction

For each removal candidate, we calculate 25 different features which are supposed to describe its complete grammatical structure. Some of the features are obtained using the Stanford coreNLP tool. We explored four different categories of features: based on statistical information, dependency and constituency parsing, Name Entity Recognition (NER), and Part-Of-Speech (POS) tagging. The full list of features is as follows.

1. **General** quantitative features

length: Number of tokens in the subtree.

ratio: Number of tokens in the subtree, normalized by the sentence length.

rel_start, *rel_end*: Relative location of the first/last subtree word in a sentence.

chars: Total number of characters in the subtree's words.

2. **Syntax** features²

dependency: Grammatical relation of the subtree's parent to the subtree root in the dependency parsing tree. In order to avoid overfitting, we group all dependency relations to nine groups: "core", "noncore", "spec", "noun", "comp", "coord", "case", "loose", and "other".

depth: Distance between the subtree and the root of the dependency syntax tree.

parent: Part of speech (POS) of the parent of the subtree in the dependency tree.

subj: Does a subtree contain a subject? (true / false)

verb: Does a subtree contain a verb? (true / false)

obj: Does a subtree contain an object? (true / false)

phrase_type: Phrase type of the subtree according to the sentence constituency syntax tree.³ *punct_in*: Does a subtree contain punctuation marks? (true / false)

punct_out: Is a subtree surrounded by punctuation marks? (true / false)

3. **NER** features

is_ner: is a subtree a named entity (NE)?

ner_type: NE type of a subtree.⁴

4. **POS** features

POS_before, *POS_after*: POS label of the last word preceding and the first word following the word sequence represented by the subtree.

POS_first, *POS_last*: POS label of the first/last word in the word sequence.

%nouns, *%verbs*, *%adjectives*, *%adverbs*, *%prepos*: Percentage of nouns, verbs, adjectives, adverbs, or prepositions in the subtree, respectively.

Feature selection is performed before applying a classification algorithm to the labeled data (see the next section).

2.3 Compression Algorithm

Algorithm 1 contains the pseudocode of the proposed method. The inputs of the algorithm are a set of complete sentences S to be compressed, a classification algorithm

² All syntax features are calculated using the dependency syntax tree, except for the last one, which is obtained from the constituency parse tree.

³ We use "LEAF" type for a single word

⁴ We use "null" for the non-NEs

$Algo$, and a manually annotated dataset D (with already selected features). The output of the algorithm is a set of compressed sentences C . First, a classification model M is induced from the dataset D using the algorithm $Algo$. For each sentence S_i from the set S , a dependency parse tree is created (using Stanford parser) and all subtrees of the dependency tree are identified, filtered (determiners, preposition, "wh"-words, etc.), and saved in the set of removal candidates ST_i . Every subtree (removal candidate) is converted to a sequence of words, the features (depending on the classification algorithm $Algo$) are extracted, and the model M is used to classify this candidate. All candidates that are classified as "Y" by M are removed from the corresponding sentence and the compressed sentences are added to the output set C .

For example, if we run the algorithm on single sentence "A woman was injured by a falling tree in the Gresham neighborhood", it detects (classify to "Y") the following removing candidates: "by a falling tree in the Gresham neighborhood", "in the Gresham neighborhood", "falling".

Algorithm 1: Sentence compression

Input: set S of original sentences S_1, \dots, S_n
classification algorithm $Algo$
annotated dataset D

Output: set C of compressed sentences

$M \leftarrow induce_model(D, Algo)$
 $C \leftarrow \emptyset$

foreach $S_i \in S$ **do**

$DT_i \leftarrow generate_dep_tree(S_i)$
 $ST_i \leftarrow all_subtrees(DT_i)$
 $ST_i \leftarrow filter(ST_i)$
 $Cands_i \leftarrow \emptyset$

foreach $ST_{ij} \in ST_i$ **do**

$Cand_{ij} \leftarrow get_words(ST_{ij})$
 $Attr_{ij} \leftarrow attributes(Cand_{ij})$
 $Class \leftarrow classify(Attr_{ij}, M)$
if $Class = Y$ **then**
padding-left: 60px; $Cands_i \leftarrow Cands_i \cup Cand_{ij}$
end

end
 $C_i \leftarrow S_i$
foreach $Cand_{ij} \in Cands_i$ **do**
padding-left: 40px; $C_i \leftarrow C_i - Cand_{ij}$
end
 $C \leftarrow C \cup C_i$

end
return C

All stages of our algorithm, except classification, are polynomial (quadratic) in the dependency tree size (number of its nodes). Practically, the most computationally ex-

pensive part of our compression pipeline is classifying the removal candidates by a specific classification model (e.g., SMO-NPK).

3 Experiments

We performed both automated and human evaluations of the proposed method. Automated evaluations were aimed at selecting the most accurate subtree classifier for our compression procedure. The purpose of human evaluations was to estimate the quality of compression and compare our approach to the state-of-the-art method [7].

3.1 Training data

Our training data is composed of a random sample of 100 sentences from DUC 2002⁵, with various length, number of clauses, and amount of punctuation marks. Four annotators labeled the extracted removal candidates as “valid” if the corresponding word sequences could be removed without compromising the sentence meaning and its grammatical correctness. Finally, we labeled with “Y” the subtrees marked by majority of annotators as “valid” and with “N” all other subtrees. The resulting training dataset contains 1494 subtrees (690 “Y” and 804 “N” instances) extracted from 100 sentences.

3.2 Classifiers

Using the Weka tool [9], we evaluated 14 different classification algorithms and finally chose four classifiers that performed best with our dataset, namely: Logistic Regression (LR), Sequential Minimal Optimization with Normalized Poly Kernel (SMO-NPK), Classification Via Regression (CVR), and Logistic Model Trees (LMT).

Table 2 shows the subtree classification performance, measured by the precision and the recall of the “Y” class using 10-fold cross validation and all 25 features.

Classifier	Precision	Recall	F 0.5
LR	0.750	0.754	0.751
SMO-NPK	0.766	0.741	0.761
CVR	0.739	0.761	0.743
LMT	0.737	0.771	0.744

Table 2. Evaluation of various classifiers.

In order to simplify the trained models, shorten training times, and reduce overfitting, we performed feature selection with the following methods: iterative subset selection with Gain Ratio and Info Gain (bottom-up) and backward elimination with “Y” precision (top-down).

Performance of feature selection methods is shown in Figures 3, 4 and 5.

⁵ <http://www-nlpir.nist.gov/projects/duc/data.html>

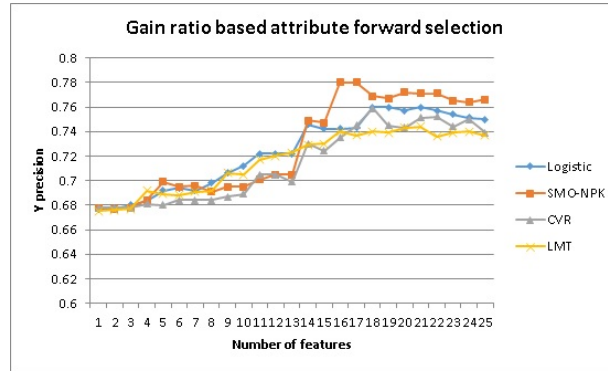


Fig. 3. Attribute selection using Gain Ratio.

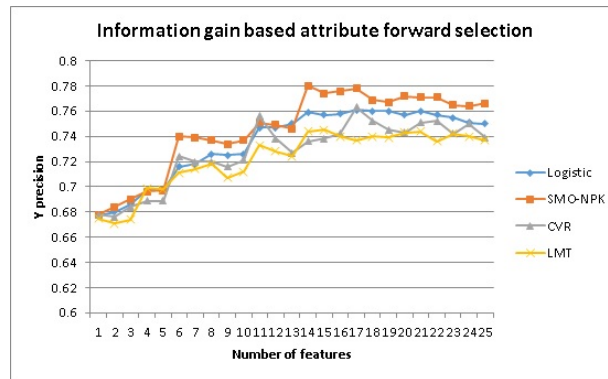


Fig. 4. Attribute selection using Info Gain.

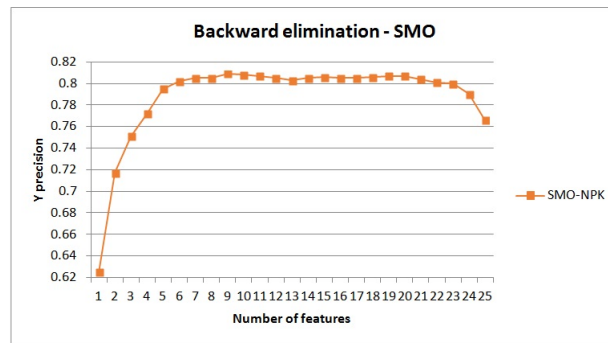


Fig. 5. Backward Elimination using “Y” precision.

The best “Y” precision and recall values obtained by each one of feature selection methods are shown in Table 3 and Table 4, respectively. The best “Y” class precision

(0.802) was achieved using backward elimination with SMO classifier. The following six features were finally selected: *parent*, *punct_out*, *POS_before*, *POS_after*, *POS_first*, and *POS_last*. With the exception of *punct_out*, all these features are based on POS tags of specific words in the subtree and in the main sentence.

Method	LR	SMO-NPK	CVR	LMT
All (25) features)	0.750	0.766	0.739	0.737
Gain Ratio	0.760	0.780	0.752	0.744
Info Gain	0.761	0.780	0.763	0.745
Backward Elimination	-	0.802	-	-

Table 3. "Y" class **precision** values obtained with various feature selection methods.

Method	LR	SMO-NPK	CVR	LMT
All (25) features)	0.754	0.741	0.761	0.771
Gain Ratio	0.761	0.728	0.759	0.768
Info Gain	0.767	0.735	0.759	0.771
Backward Elimination	-	0.720	-	-

Table 4. "Y" class **recall** values obtained with various feature selection methods.

Method	LR	SMO-NPK	CVR	LMT
All (25) features)	0.751	0.761	0.743	0.744
Gain Ratio	0.760	0.769	0.753	0.749
Info Gain	0.762	0.771	0.762	0.750
Backward Elimination	-	0.784	-	-

Table 5. "Y" class **F 0.5 measure** values obtained with various feature selection methods.

Different number of features selected by each one of the feature selection methods. According to forward selection with the Gain Ratio and the Info Gain measures, "dependency" is the most important feature having the highest Gain Ratio and Info Gain values, which implies that the subtree's dependency relation to the main sentence should be the leading criterion for choosing removal candidates in a dependency tree. However, SMO-NPK provided a better "Y" precision (0.802) than all other classifiers, without this feature and with the smallest number of features (six), when Backward Elimination was used.

3.3 Test Data

For human evaluation, we use the first 200 sentences from a dataset available online and used in [7]. The compression ratio is defined by the number of characters in compressed

sentences divided by the number of characters in the original sentences. We applied our method to each sentence several times repeatedly, with the first run processing the original sentence from the dataset and each subsequent run applied to the sentence compressed by the preceding run. As expected, each subsequent run obtained a higher compression ratio that converged to some value after four iterations. The compression ratios (CR) for iterations 1, 2 and 4 as well as for the LSTM-based method from [7] are shown in Table 6.

3.4 Human Evaluation

We evaluated LSTM and the first, second and fourth runs of our method in an experiment with human raters. The raters were eight graduate students not involved otherwise in this research project. The raters were asked to rate the readability and the informativeness of generated compressions. Readability measures the grammatical correctness, comprehensibility and fluency of the output whereas informativeness measures the amount of important content preserved in the compression. Every sentence-compression pair was rated by two raters who were asked to select a score on a five-point scale. The results are shown in Table 6. The results show that our mean compression ratio is much higher than in LSTM (resulting in higher informativeness) whereas the average readability of our compressions is lower by 10% only (p value < 0.0001 for readability between our first run and LSTM). Taking into account that our model was trained on a very small dataset of only 100 sentences vs. two million in [7], our initial results are quite encouraging. Lower readability compared to LSTM may be explained by a very small training dataset and a relatively low precision of the classification model (80% only), meaning that in 20% of cases we may remove a wrong subtree and produce a grammatically incorrect sentence.

Method	Readability	Information	CR
Run1	4.06	3.47	0.63
Run2	3.72	2.99	0.55
Run4	3.44	2.77	0.52
LSTM	4.49	3.32	0.39

Table 6. Readability, informativeness and compression ratio of first, second and fourth iterations of our compression method compared to the LSTM .

4 Conclusions

We implemented a sentence compression tool that uses the subtrees of the dependency parse trees of the original sentences as potential removal candidates and calculates 25 predictive features for each subtree. To train a supervised learning algorithm we created a small dataset based on subtrees labeled by human annotators. We experimented with several classification algorithms and found that SMO (Normalized Poly Kernel) performed best with our dataset requiring only 6 out of 25 features (mostly based on POS

tags) to classify the removal candidates. Subsequently, we utilized the SMO classifier for choosing subtrees to be removed by our sentence compression pipeline and used it to compress 200 sentences from a separate benchmark dataset. The results of the human evaluation indicate that using a very small training dataset of only 100 sentences, our compressions are only slightly less readable than the compressions produced by the LSTM algorithm, which was trained on two million sentence-compression instances. Also, our pipeline can be easily adapted to other languages given a sentence splitter, tokenizer, dependency parser, and POS tagger on those languages.

References

1. Berg-Kirkpatrick, T., Gillick, D., Klein, D.: Jointly learning to extract and compress. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. pp. 481–490. Association for Computational Linguistics (2011)
2. Clarke, J., Lapata, M.: Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research* 31, 399–429 (2008)
3. Cohn, T., Lapata, M.: Large margin synchronous generation and its application to sentence compression. In: EMNLP-CoNLL. pp. 73–82 (2007)
4. Cohn, T., Lapata, M.: Sentence compression beyond word deletion. In: Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1. pp. 137–144. Association for Computational Linguistics (2008)
5. Colmenares, C.A., Litvak, M., Mantrach, A., Silvestri, F.: HEADS: Headline generation as sequence prediction using an abstract feature-rich space. In: North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL HLT 2015). pp. 133–142 (2015)
6. Corston-Oliver, S.: Text compaction for display on very small screens. In: Proceedings of the NAACL Workshop on Automatic Summarization. pp. 89–98. Citeseer (2001)
7. Filippova, K., Alfonseca, E., Colmenares, C.A., Kaiser, L., Vinyals, O.: Sentence compression by deletion with LSTMs. In: EMNLP. pp. 360–368 (2015)
8. Filippova, K., Strube, M.: Dependency tree based sentence compression. In: Proceedings of the Fifth International Natural Language Generation Conference. pp. 25–32. Association for Computational Linguistics (2008)
9. Frank, E., Hall, M.A., Witten, I.H.: The WEKA workbench. online appendix for “data mining: Practical machine learning tools and techniques”. Morgan Kaufmann, Fourth Edition (2016)
10. Galley, M., McKeown, K.: Lexicalized markov grammars for sentence compression. In: HLT-NAACL. pp. 180–187 (2007)
11. Jing, H.: Sentence reduction for automatic text summarization. In: Proceedings of the sixth conference on Applied natural language processing. pp. 310–315. Association for Computational Linguistics (2000)
12. Knight, K., Marcu, D.: Statistics-based summarization-step one: Sentence compression. *AAAI/IAAI 2000*, 703–710 (2000)
13. Knight, K., Marcu, D.: Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence* 139(1), 91–107 (2002)
14. Lin, C.Y.: Improving summarization performance by sentence compression: a pilot study. In: Proceedings of the sixth international workshop on Information retrieval with Asian languages-Volume 11. pp. 1–8. Association for Computational Linguistics (2003)

15. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The stanford corenlp natural language processing toolkit. In: ACL (System Demonstrations). pp. 55–60 (2014)
16. McDonald, R.T.: Discriminative sentence compression with soft syntactic evidence. In: Eacl (2006)
17. Sidhaye, P., Cheung, J.C.K.: Indicative tweet generation: An extractive summarization problem? In: EMNLP. pp. 138–147 (2015)
18. Vandeghinste, V., Pan, Y.: Sentence compression for automated subtitling: A hybrid approach. In: Proceedings of the ACL workshop on Text Summarization. pp. 89–95 (2004)