Using Artificial Neural Networks to Derive Process Model Activity Labels from Process Descriptions

Mirco Pyrtek[1,2], Philip Hake[1,2], and Peter Loos[1,2]

[1] German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany;
[2] Saarland University, Saarbrücken, Germany
{mirco.pyrtek, philip.hake, peter.loos}@dfki.de

Abstract. Recently, Artificial Neural Networks (ANN) have shown high potential in the area of Natural Language Processing (NLP). In the area of sentence compression, the application of ANNs has proven to outperform existing rule-based approaches. Nevertheless, these approaches require a decent amount of training data to achieve high accuracy. In this work, we aim at employing ANNs to derive process model labels from process descriptions. Since the amount of publicly available pairs of text and process model is scarce, we employ a transfer learning approach. While training the compression model on a large corpus consisting of sentence-compression pairs, we transfer the model to the problem of deriving label descriptions. We implement our approach and conduct an experimental evaluation using pairs of process descriptions and models. We found that our transfer learning model keeps high recall while losing performance on precision and compression rate.

Keywords: Business Process Modeling, Deep Leaning, Sentence Compression, Artificial Neural Network, Natural Language Processing

1 Introduction

Business process modeling is an important task of Business Process Management (BPM) [1]. It is used in manifold applications including the standardization and the improvement of business processes [2]. Process modeling requires the knowledge of a modeling language and the underlying process domain. A process modeler may acquire the latter by interviews, observations and investigating process descriptions. However, gathering the process requirements and deriving process models represent an effortful cognitive performance. Thus, first approaches investigate an automated analysis [3] of process descriptions and the automated derivation of process models [4]. Existing approaches are rule-based and built up on established methods from Natural Language Processing (NLP) to extract the information contained in the descriptions. However, since process descriptions represent unstructured information, the automated analysis and derivation of process models becomes easily complex.
Recently, deep learning approaches have been proven to solve complex cognitive tasks such as language translation [5] and compression of information [6]. Furthermore, deep

15th International Conference on Wirtschaftsinformatik,
March 08-11, 2020, Potsdam, Germany

learning-based classification has been successfully applied in the context of BPM to automatically distinguish activity from event labeling in process models [7].
Due to the promising results of deep learning in NLP, we aim at investigating the potential of deep learning-based NLP for the automated translation from process descriptions to conceptual process models. In particular, we focus on the research question (RQ) how sentences of process description can be condensed to create activity labels that retain the most important process information. Thus, in this work we investigate sentence compression [8] as a technique to mimic the labelling task of a process modeler. In the past, sentence compression has been used in variety of applications, including compression of spoken sentences in closed captioning services for deaf persons watching TV [9] and generating abstracts for documents [10].
Nevertheless, deep learning-based compression requires a large data set containing pairs of sentences and their respective compression. To our best knowledge, a data set containing a sufficient number of sentences of process descriptions and their respective labels is not publicly available. Thus, we aim at employing a transfer learning approach to overcome the lack of domain specific training data [11]. To answer the research question, we use a design science research approach [12]. We conceptualize a compression model as our artifact of interest, implement the compression model and provide an evaluation of our compression approach.

The paper is structured as follows. Section 2 presents related work. In section 3, we provide a detailed description of the conceptual design of our developed deep learning-based compression model. Section 4 presents the evaluation of our approach including a detailed description of the data as well as the evaluation setup. Finally, section 5 discusses the achieved results, the limitations of our approach and future work.

2 Related Work

In BPM, approaches aiming at automatic derivation of conceptual models from text can be found frequently [4]. Friedrich, et al. [4] makes use of methods from computational linguistics and NLP in order to extract process models from textual descriptions. Doing so, they defined a set of rules and markers extracted by a syntax parser to identify relevant activities in a sentence. However, such rule-based approaches heavily rely on correctly parsed syntax trees and, therefore, are very error-prone specifically when having noisy input data [4].

Thus, we investigated another method called sentence compression to extract relevant information i.e. activity labels from text. There are mainly two different methods to solve a compression task of a sentence. While *abstractive* approaches [13-14] rely on paraphrasing words, *extractive* methods [6], [8-9], [15-20] solve sentence compression as a sequence of word deletions of the original sentence. In this case, for each word of a sentence, the compression algorithm needs to decide whether to keep or delete the word based on its given features [17]. In deletion-based sentence compression, one can distinguish between two different lines of research: one relying on manually modeled linguistic knowledge [15] and the other based on machine learning (ML) [18].

In linguistic-knowledge-based sentence compression approaches make use of syntactic features as signals [21-23]. Following the approach of syntactic features, the task of sentence compression is defined as an optimization problem using hard constraints, which can be solved by integer linear programming (ILP) [24]. Other work in this line focuses on pruning dependency trees in order to shorten a sentence [8], [25-26]. Similar to rule-based model derivation in BPM, compression models solely based on linguistic-knowledge are highly sensitive to errors since there is no way to recover from an incorrect parse tree [15].

Therefore, the second research stream in deletion-based sentence compression employs methods from ML and deep learning resulting in increased performance. Filippova, et al. [15] use a Long-short term memory (LSTM) [27] model to compress a sentence by using word embedding without any additional syntactical information. A similar approach is described by Sakti, et al. [9] using incremental sentence compression, i.e. deciding at each step of the sequence whether to keep or remove the current word instead of processing the entire sequence at once.

The deletion-based ML approaches are further investigated in the following research. Wang, et al. [17] use syntactical features in addition to word embeddings. Lai, et al. [16] propose a bi-directional encoder-decoder approach while Thao, et al. [18] use a gated neural network. ML models dedicated to classifying the word tokens of a sentence that will be omitted are also investigated in [19]. The approach aims at applying sentence compression in a cross-lingual setting learning from sentences in two different languages, English and Portuguese. Further approaches make use of unsupervised learning to train sentence compression models. Miao and Blunsom [20] trained a generative model using competitive generator and discriminator networks to generate compressed sentences. Wang, et al. [6] formulated sentence compression as a Markov decision process and reinforcement learning is applied in order to train a neural network learning syntactical constraints known from previous ILP research on sentence compression in order to compress a sentence properly.

3 Conceptual Design

In this work, we design an artifact that is able to derive process model activity labels from textual process descriptions. In this section, we describe the conceptual design of our approach. We aim at training a deletion-based compression model [15] that is able to tag each word of a sentence whether it should be omitted or not. In other words, for a sentence $s = [w_1, w_2, \dots, w_n]$ there exists a corresponding sequence of integers $y = [y_1, y_2, \dots, y_n]$ with $y_i \in [0,1], i \in [1, \dots, n]$ [17]. In this context, $y_i = 1$ means the word is part of the compression and $y_i = 0$ means it is not.

In Figure 1 the design process of our artifact is illustrated. The process starts with the underlying data sources extracted from two different domains. The first corpus contains 210,000 headlines of English news articles together with their appropriate

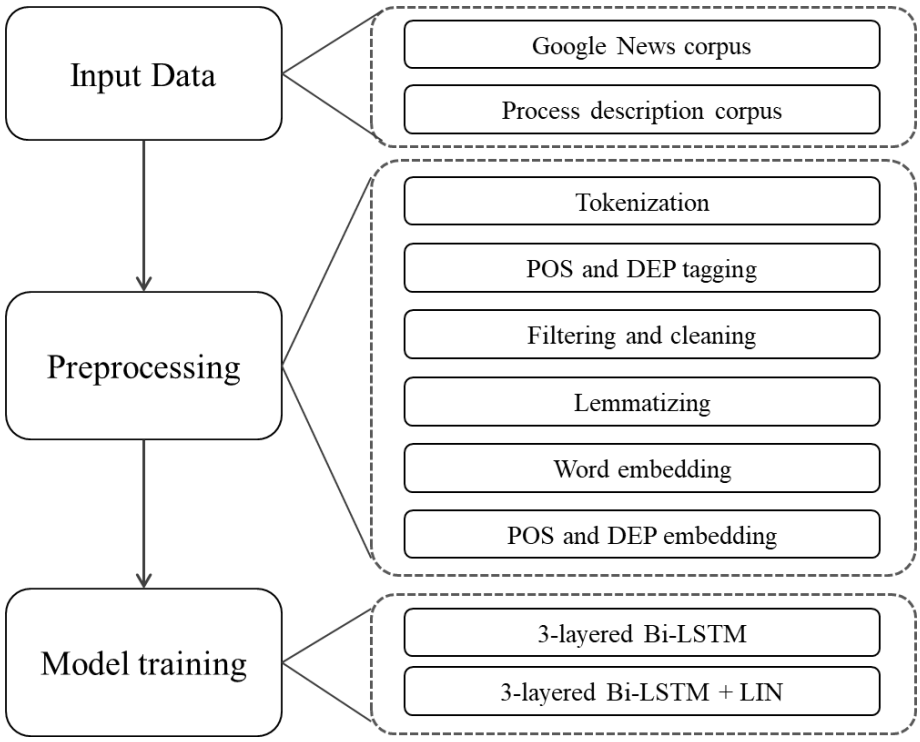

Figure 1. Conceptual Design

compressions and was collected and used by Filippova and Altun [26]. Our second dataset includes three different process descriptions manually collected by us given in text form with a total of 32 sentences. Next to this, we continue with a preprocessing of the raw data using techniques known from NLP research as well as predictive modeling. The stage of preprocessing includes all steps necessary to transform a sentence into a sequence of feature vectors.

First, the raw sentences and their compressions need to be parsed using a syntax parser. We tokenize the sentences into sequences of words and gather syntactical information, which will be used by our linguistic knowledge model. The purpose of tokenization is transforming the raw sentence-compression-pairs into labeled data for the model training defining. Based on the compressed sentence each token of the original sentence is assigned a tag indicating whether it should be part of the compression or not. Furthermore, syntactical information is attached to each token, including dependency (DEP) labels and part-of-speech (POS) tags.

Subsequently, all digits, dates and numbers are replaced by a special NUMB token in order to simplify the sentences by deleting irrelevant information [28]. For further simplification, each sentence is filtered by all tokens that are punctuation marks. Excluded from the filtering process are the punctuation marks at the end of each sentence. These tokens are not deleted but transformed into end-of-sequence (EOS)

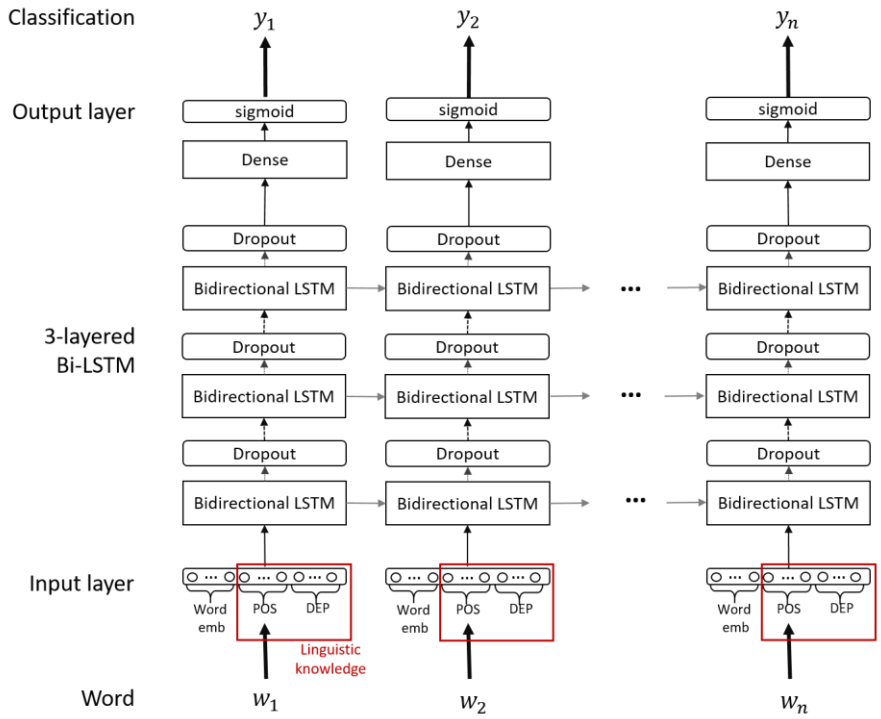

Figure 2. Model architecture according to Wang, et al. [17]

tags following the approach of Filippova, et al. [15]. The process of filtering punctuation marks is motivated by the reduction of unique POS. Next to this we apply a stemming algorithm for reducing the amount of unique words sharing the same meaning.

After these steps of preprocessing the words are finally transformed into embedded feature vectors that can be processed by neural networks. Therefore, a standard skip-gram model is employed in order to transform each word of the vocabulary into a d-dimensional vector [18]. Furthermore, the categorical features for a token (POS tags and DEP labels) are encoded using one-hot-encoding. Because of the filtering of punctuation marks, the vector size of this encoding can be reduced in this step resulting in an increased training performance. The preprocessed data is then used for training our compression model and deriving labels from textual process descriptions after the training.

We investigate a base model and a Linguistic Knowledge (LK) model in this work. In Figure 2, the architecture of both models is depicted. As illustrated, both models share the same architecture but differ in their corresponding input feature space. The base model solely utilizes word embeddings (Word emb) as input features whereas the LK model uses word embeddings with added linguistic knowledge (POS and DEP). The input layer processes a set of features for each word that, depending on the model,

incorporates linguistic knowledge or solely word embeddings. As already mentioned above, LK considers two kinds of linguistic features for the model as input features, DEP labels and POS tags.

The architecture and model parameters mainly follow the approach of Wang, et al. [17]. The model is trained using the architecture of a 3-layered bi-directional LSTM followed by a dropout layer to avoid overfitting. Each LSTM cell contains 123 neurons resulting in 256 activations per feature vector due to the bi directionality of our model. Each dropout layer is initialized with a fraction of 0.5. The last LSTM layer is connected to a fully connected dense. Finally, we apply a sigmoid activation function on the activations of the dense layer resulting in a binary classification output for each processed word. This classification y_i is considered the prediction of our model on whether the current word is part of the compression or not.

4 Evaluation

4.1 Data and Preprocessing

As part of our design, we make use of two independent data sets that are extracted from two different domains, English news headlines and text-based business process descriptions. The first one is used for both, training and evaluation of the model, and contains sentences together with their corresponding compression. To give an example for a sentence extracted from the news corpus the headline "The Australian Treasury believes positive signs are emerging in the Australian economy" is denoted together with its compression "Positive signs are emerging". This data is freely available in a public repository.[1]

The latter data set is incorporated during evaluation only and contains the process' text descriptions. The texts were initially given in German language and were automatically translated into English using a machine translation tool.[2]

As already discussed in the conceptual design section earlier, the data first needs to run through several steps of preprocessing to transform the sentences given as strings into a sequence of feature vectors. We use syntaxnet[3] as our underlying syntax parser for transforming sentences into sequences of words and gathering DEP labels and POS tags. During this step of preprocessing a few sentence-compression pairs are dropped because of a different order of the words of the sentence and its corresponding compression, wrong encoding or extensive length of the sentence. Subsequently, we apply a WordNet [29] lemmatizer using python's nltk package[4] for stemming of the raw words. Furthermore, we employ Word2Vec [30] for embedding of the words into feature vectors using python's genism package[5]. We initialize d = 200 as the dimensionality for the embedding vectors and train the embedding model using the

[1] https://github.com/google-research-datasets/sentence-compression
[2] https://www.deepl.com/translator
[3] https://github.com/tensorflow/models/tree/master/research/syntaxnet
[4] https://www.nltk.org/_modules/nltk/stem/wordnet.html
[5] https://radimrehurek.com/gensim/

words from both datasets. Finally, categorical DEP and POS tags are transformed into feature vectors using one-hot-encoding.

All the relevant statistics for both data sets after preprocessing can be found in Table 1 with S denoting the complete set of sentences. In this table the number of sentences, vocabulary size, number of unique POS and DEP classes, average sequence lengths, as well as compression rates are described. Finally, we apply padding to the sequences such that they stick to the internal shape of a deep recurrent neural network.

Table 1. Dataset statistics

	News Data	**Process Data**
$\|\mathbf{S}\|$	200 969	32
$\|\mathbf{S^{voc}}\|$	107 449	199
$\|\mathbf{S^{POS}}\|$	49	26
$\|\mathbf{S^{DEP}}\|$	43	29
$\mathbf{avg}(\|\mathbf{S_i}\|)$	26.38	17.28
Compression rate	0.3917	0.4268

4.2 Model Training

The model is compiled using Adam [31] as the underlying optimizer with a learning rate of 0.001. As the loss function the negative log-likelihood function is used. We train both models over 10 epochs using a batch size of 32.

The models are implemented in python, version 3.6, constructed and compiled using the deep learning library keras[6], version 2.2.4, running TensorFlow[7], version 1.11.0, in the backend.

We split the news corpus is into training, test and validation data sets. We keep 1.000 sentences for each, test and validation data set and the rest of 198.969 for training. The ratio between the training, test and validation set is set following the approach of Filippova, et al. [15].

4.3 Performance Measures

We use precision, recall, F1-score and accuracy as defined in formulas (1)-(4) to measuring the model's performance. In this context, we compare predicted deletion marks to the true labels given in the test data. The true positives (TP) represent the labels that are correctly predicted as 1, the true negatives (TN) are all labels correctly predicted as 0, the false positives (FP) are predicted as 1 but in fact are labelled with 0 and the false negatives (FN) are predicted as 0 but in fact are labelled with 1.

[6] https://keras.io/

[7] https://www.tensorflow.org/

$$precision = \frac{TP}{TP + FP} \tag{1}$$

$$recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{3}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Furthermore, we calculate the compression rate (CR). The CR defines the average ratio between the number of words being part of the compression and the number of words in the original sentence. As defined in formula (5), we calculate this ratio for all sentences y containing labelled word embeddings y_i. In other words, CR denotes the average rate of words to keep of a sentence of certain length when compressing it.

$$\mathrm{CR(y)} = \frac{|\{y_i | y_i \in y, y_i = 1\}|}{|\{y_i | y_i \in y\}|} \tag{5}$$

4.4 Results

Finally, in this section the results of our approach are evaluated. After training the models, they are evaluated on unseen data of the news corpus as well as the process descriptions dataset by applying each one on the test data and calculating the performance measures for each of them. Since labels are only given for the news data, we have to assign true labels for the process descriptions manually. In this process, the words in the compression are kept following order and syntax of the initial sentence in the process description because of our deletion-based approach. Thus, sentences formulated passively were also formulated passively in the related compression.

Table 2. Performance measures

	Model	Precision	Recall	F1	acc_{word}	$acc_{sentence}$
News Data	**Base Model**	0.7832	0.7486	0.7655	0.8204	0.1100
	LK Model	**0.8423**	**0.8024**	**0.8219**	**0.8638**	**0.2320**
Process Data	**Base Model**	0.5797	0.7155	0.6404	0.6528	**0.0625**
	LK Model	**0.6471**	**0.7364**	**0.6888**	**0.7125**	0.0313

In Table 2 the scores of both models applied on the test data can be found. As we can see from here, two different types of accuracy measures are calculated. We investigate a per word accuracy and a per sentence accuracy giving a measure for whole sequences of words in a sentence to be correctly classified [28]. We can observe that the LK model achieves overall better results w.r.t. the metrics except per sentence accuracy of the process data. For the news corpus, the incorporation of linguistic knowledge empowers the model to classify as double as much sentences correctly compared to the base model. However, his rule does not hold for the process data. The reason for that could be explained by the different grammatical syntax of both datasets.

Furthermore, we can see that for the process data, we achieve a relatively high recall value of 0.7364 although the process data is not incorporated during training of the model. The model seems to identify all important terms within the sentences with rather high accuracy for both datasets. In contrary, precision values significantly drop for both models when applied on the process descriptions. This means that the models are keeping a lot of unnecessary words in the compressed label which could be due to the different grammar and vocabulary of both datasets.

In Table 3 the particular CR of the ground truth i.e. of the test set and of the particular model outputs are denoted. As one can see from the table, the CR differ significantly for the process data. The models are keeping much more words in the compression when applied on the processes compared to the news corpus. Obviously, the model produces more false positives, since the real CR of the dataset is significantly lower. This is also reflected in the low precision score of both models. However, incorporating process data during model training can possibly improve both values, CR and precision, since specific syntax and vocabulary of this data can be taught.

Table 3. Compression rates

Data	**Compression Rate**	
	News Data	**Process Data**
Test data	0.3915	0.4268
Base model	0.3742	0.5335
LK model	0.3729	0.4919

5 Discussion

Our compression model was found to keep high recall scores when applied to the process data but is limited to lower precision scores compared to the news corpus. This results in compressions, which are slightly longer than the true activity labels and require manual post-processing. Thus, improvement of the model can possibly be achieved by applying techniques of on-line or incremental learning as described by Losing, et al. [32]. Following such approach, domain-specific data is added to the training data set including process descriptions and according compressions, which allows the model to learn syntax and vocabulary that are specific to this domain. However, this requires additional work of data gathering since such data is not publicly

available. Moreover, the proposed models are limited to a sentence-based processing. Thus, information spread over multiple sentences cannot be summarized appropriately. Furthermore, the investigation of the trained models revealed that they are limited to preserving only single activities within a sentence of a process description. For instance, a single sentence of a process description could consist of multiple main or subordinate clauses containing several process activities. In this case, single activities cannot be identified separately by the compression model resulting in incorrect or missing activity labels. This problem originates from the applied compression in the training dataset where the compressed sentences mainly exhibit a single activity. This issue could be resolved by parsing and splitting the sentences using a syntax parser before passing the clauses to the model.

6 Conclusion

In this work, we investigated the RQ how sentences of process description can be condensed to create activity labels that retain the most important process information. We applied a design science research approach resulting in a sentence compression model based on recurrent neural networks as our artifact of interest. Furthermore, we employed a transfer learning approach to overcome the lack of publicly available pairs of process models and process descriptions. We found that our trained compression model is able to keep significantly high recall while losing performance on precision as well as compression rates.

In Table 4 we provide two examples for output compressions by the model denoted together with the original sentence and the true label. As already indicated by the high recall score, both examples fully contain the words of the real label, whereas they also carry additional words that are actually not part of it. Since the amount of information covered by the compressed sentence is ~50% less on average (according to CR) compared to the original one while keeping most important words, the compression of the original sentence can possibly reduce the amount of cognitive effort required for a process modeler to extract the right label manually from a process description. In particular, a process modeler can utilize our model to get recommendations for activity labels of process models based on its textual descriptions. In future work, the impact of the model w.r.t. time savings for a process modeler can be analyzed in a user study.

Table 4. Example output

Sentence	Label	Compression (LK model)
the goods are delivered by the vendor and arrive in the warehouse	goods are delivered	the goods are delivered by the vendor
if the latter is the case the goods must be repacked to make them fit for storage	goods must be repacked	the goods must be repacked to make fit for storage

References

1. C. Houy, P. Fettke, P. Loos, W. van der Aalst und J. Krogstie, „Business Process Management in the Large," Business & Information Systems Engineering, pp. 385-388, June 2011.
2. M. Malinova und J. Mendling, „A Qualitative Research Perspective on BPM Adoption and the Pitfalls of Business Process Modeling," in International Conference on Business Process Management, Tallinn, Estonia, 2012.
3. H. van der Aa, H. Leopold und H. Reijers, „Comparing textual descriptions to process models–the automatic detection of inconsistencies," in Information Systems, 2016.
4. F. Friedrich, J. Mendling und F. Puhlmann, „Process model generation from natural language text," in International Conference on Advanced Information Systems Engineering, Berlin, Heidelberg, 2011.
5. M. X. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G. Foster, L. Jones, N. Parmar, N. Shazeer, A. Vaswani, J. Uszkoreit, L. Kaiser, M. Schuster, Z. Chen, Y. Wu und M. Hughes, „The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation," in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 2018.
6. L. Wang, J. Jiang und L. Liao, „Sentence Compression with Reinforcement Learning," in International Conference on Knowledge Science, Engineering and Management, Cham, 2018.
7. P. Hake, M. Zapp, P. Fettke und P. Loos, „Supporting Business Process Modeling Using RNNs for Label Classification," in International Conference on Applications of Natural Language to Information Systems, Liège, 2017.
8. K. Knight und D. Marcu, „Statistics-Based Summarization - Step One: Sentence Compression," in Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, 2000.
9. S. Sakti, F. Ilham, G. Neubig, T. Toda, A. Purwarianti und S. Nakamura, „Incremental sentence compression using LSTM recurrent networks," in Automatic Speech Recognition and Understanding, 2015.
10. K. Knight und D. Marcu, „Summarization beyond sentence extraction: A probabilistic approach to sentence compression," Artificial Intelligence, pp. 91-107, 1 July 2002.
11. S. J. Pan und Q. Yang, „A survey on transfer learning," IEEE Transactions on knowledge and data engineering, pp. 1345-1359, 2009.
12. K. Peffers, T. Tuunanen, C. Gengler, M. Rossi, W. Hui, V. Virtanen und J. Bragge, „The design science research process: A model for producing and presenting information systems research," in Proceedings of First International Conference on Design Science Research in Information Systems and Technology DESRIST, Claremont, California, 2006.
13. S. Chopra, M. Auli und A. M. Rush, „Abstractive Sentence Summarization with Attentive Recurrent Neural Networks," Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 93-98, 1 January 2016.
14. T. Cohn und M. Cohn, „Sentence Compression Beyond Word Deletion," 22nd International Conference on Computational Linguistics, pp. 137-144, 2008.
15. K. Filippova, E. Alfonseca, C. Colmenares, L. Kaiser und O. Vinyals, „Sentence compression by deletion with LSTMs," in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015.

16. D.-V. Lai, N. T. Son und N. Le Minh, „Deletion-based sentence compression using Bi-enc-dec LSTM," in International Conference of the Pacific Association for Computational Linguistics, Singapore, 2017.
17. L. Wang, J. Jiang, H. L. Chieu, C. H. Ong, D. Song und L. Liao, „Can Syntax Help? Improving an LSTM-based Sentence Compression Model for New Domains," in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, 2017.
18. Y. Zhao, H. Senuma, X. Shen und A. Aizawa, „Gated Neural Network for Sentence Compression Using Linguistic Knowledge," in International Conference on Applications of Natural Language to Information Systems, 2017.
19. F. A. A. Nóbrega und T. A. S. Pardo, „Investigating Machine Learning Approaches for Sentence Compression in Different Application Contexts for Portuguese," in International Conference on Computational Processing of the Portuguese Language, Cham, 2016.
20. Y. Miao und P. Blunsom, „Language as a Latent Variable: Discrete Generative Models for Sentence Compression," in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, Texas, 2016.
21. J. Bingel und A. Søgaard , „Text Simplification as Tree Labeling," in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) , Berlin, Germany, 2016.
22. H. Jing, „Sentence Reduction for Automatic Text Summarization," in Proceedings of the Sixth Conference on Applied Natural Language Processing, Seattle, Washington, USA, 2000.
23. R. McDonald, „Discriminative sentence compression with soft syntactic evidence," in 11th Conference of the European Chapter of the Association for Computational Linguistics, 2006.
24. J. Clarke und M. Lapata, „Global Inference for Sentence Compression an Integer Linear Programming Approach," 2008.
25. T. Berg-Kirkpatrick, D. Gillick und D. Klein, „Jointly Learning to Extract and Compress," in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, Portland, Oregon, 2011.
26. K. Filippova und Y. Altun, „Overcoming the Lack of Parallel Data in Sentence Compression," in Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing , Seattle, Washington, USA, 2013.
27. S. Hochreiter und J. Schmidhuber, „Long Short-term Memory," Neural Computation, pp. 1735-1780, 1 December 1997.
28. F. Rodrigues, B. Martins und R. Ribeiro, „Neural Methods for Cross-Lingual Sentence Compression," in International Conference on Artificial Intelligence: Methodology, Systems, and Applications, Cham, 2018.
29. G. A. Miller, „WordNet: a lexical database for English," Communications of the ACM, Bd. 38, Nr. 11, pp. 39-41, 1995.
30. T. Mikolov, K. Chen, G. Corrado und J. Dean, „Efficient Estimation of Word Representations in Vector Space," Proceedings of the International Conference on Learning Representations, pp. 1-12, 1 January 2013.
31. D. Kingma und J. Ba, „Adam: A method for stochastic optimization," in International Conference for Learning Representations, San Diego, 2015.
32. V. Losing, B. Hammer und H. Wersing, „Incremental on-line learning: A review and comparison of state of the art algorithms," Neurocomputing, pp. 1261-1274, 31 January 2018.