Fact-aware Abstractive Text Summarization using a Pointer-Generator Network

System Description for the "German Text Summarization" Challenge at SwissText 2019

Valentin Venzin
SpinningBytes AG
valentin.venzin@gmail.com

Jan Deriu
Zurich University of Applied Sciences
deri@zhaw.ch

Didier Orel
Tamedia AG
didier.orel@tamedia.ch

Mark Cieliebak
SpinningBytes AG
mc@spinningbytes.com

ABSTRACT

In this work we use a pointer-generator network for abstractive text summarization. In addition to generating tokens from a fixed vocabulary, the sequence-to-sequence model is able to copy certain passages from the source text to the output.

We identified two issues: Repeated tokens or phrases in the generated summary and fact fabrication. Based on [8], the former is addressed by modifying the attention mechanism and the loss function of the model. We try to avoid fact fabrication by also supplying the model with extracted *fact descriptions* from the article as suggested in [4].

We train the model on the 100′000 provided Wikipedia article and summary pairs. The generated summaries are in most cases relevant to the corresponding article, grammatically correct and fluent.

1 SYSTEM DESCRIPTION

Figure 1 shows a schematic overview of our system. We discuss the main components in the following subsections. articles

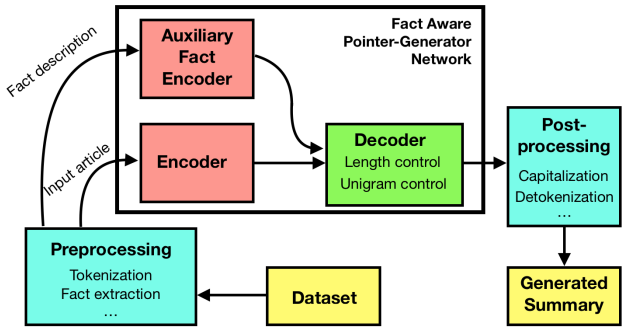

Figure 1: Fact aware pointer-generator network.

1.1 Preprocessing

Using spaCy[1] we lowercase and tokenize articles and summaries. On the tokenized corpus, we construct a vocabulary of the 60′000 most frequent tokens.

[1]https://spacy.io/

As in [8], to speed up training and testing, each article is truncated to 400 tokens. By comparison: The mean and the median number of tokens of the provided Wikipedia articles is 708.83 and 583.00, respectively. Thus, we believe that this restriction does not have a big impact on the results. Furthermore, most relevant information will arguably appear towards the beginning of the article.

1.2 Pointer-Generator Network

The core of our system is the *pointer-generator network*, based on a sequence-to-sequence model with attention, proposed by See et al. [8]. Refer to Figure 2 for an illustration. Unlike in [8], we initialize our learnable word embeddings with fastText embeddings [3] which were pretrained on Wikipedia.

Sequence-to-Sequnce Model with Attention. Let h_i and s_t be the encoder and decoder LSTM states, respectively. The *attention distribution* a^t at decoder time step t is computed as in Bahdanau et al. [2]:

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{\text{attn}}) \quad (1)$$

$$a^t = \text{softmax}(e^t) \quad (2)$$

The model is given the opportunity to learn the weights v, W_h, W_s and b_{attn} such that a^t can be used to decide where in the article 'to look' when generating the output token at time t. The attention distribution is used to calculate the *context vector* h_t^*, a weighted sum of the encoder states h_i.

$$h_t^* = \sum_i a_i^t h_i \quad (3)$$

Based on a^t, the context vector captures features of the input article at specific locations. Together with the decoder state s_t, h_t^* is used to calculate a distribution P_{vocab} over the initial vocabulary:

$$P_{\text{vocab}} = \text{softmax}(V'(V[s_t, h_t^*] + b) + b') \quad (4)$$

where V, V', b and b' are trainable parameters.

Pointer-Generator Mechanism. So far, we reviewed a standard sequence-to-sequence model with attention: P_{vocab} can be used to calculate a loss function or to decode the next word at inference time. We now briefly discuss how *pointers* can be used to directly copy some tokens from the input to the output. This includes out of vocabulary (OOV) tokens from the input article.

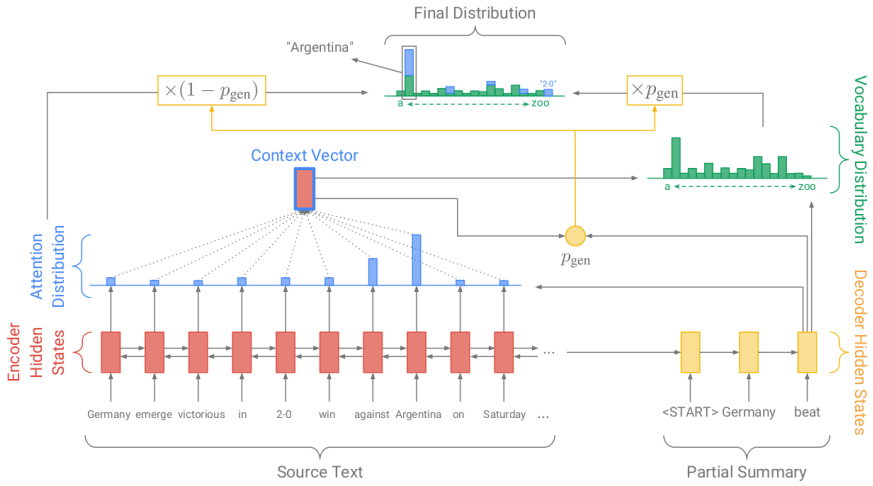

Figure 2: From [8]. The source text gets encoded by a bidirectional LSTM and decoded by an unidirection LSTM. At each decoding step t, an attention distribution a^t is computed. From a^t, we derive the pointer distribution $P_{\textbf{pointer}}$ by summing up the probabilities of all occurrences per input token. To get the final distribution P from which tokens are generated, we add the vocabulary distribution $P_{\textbf{vocab}}$ (weighted by $p_{\textbf{gen}}$) and $P_{\textbf{pointer}}$ (weighted by $(1-p_{\textbf{gen}})$). In-article OOVs have probability 0 under $P_{\textbf{vocab}}$ and tokens that do not occur in the article have probability 0 under $P_{\textbf{pointer}}$. However, the sum of the two distribution,P, might yield strictly positive probabilities for all tokens of the extended vocabulary. One advantage of this system is that in-article OOVs can still be generated by copying tokens from the source text.

The *pointer distribution* over the tokens w_i of the input article is defined as follows:

$$P_{\text{pointer}}(w) = \sum_{i:w_i=w} a_i^t \tag{5}$$

Given an input article, we define the *extended vocabulary* to be the union of the initial vocabulary and OOV tokens in the article (The latter get temporary token ids). The *final distribution* over the extended vocabulary is a weighted sum of the vocabulary and pointer distributions.

$$P(w) = p_{\text{gen}}P_{\text{vocab}}(w) + (1-p_{\text{gen}})P_{\text{pointer}}(w) \tag{6}$$

If w does not occur in the article then $P_{\text{pointer}}(w) = 0$. If w occurs in the article but not in the vocabulary (i.e. it only occurs in the extended vocabulary) then $P_{\text{vocab}}(w) = 0$. p_{gen} is a learned:

$$p_{\text{gen}} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{\text{ptr}}) \tag{7}$$

where w_{h^*}, w_s, w_x and b_{ptr} are trainable and x_t is the decoder input at time t.

1.2.1 Coverage. Sequence-to-sequence models often generate repeated words or phrases; e.g. [7, 9]. See et al. [8] address this problem with their *coverage* mechanism. The idea is to discourage the model to attend the same locations in the input article multiple times. To do this, they define a *coverage vector*

$$c^t = \begin{cases} \sum_{t'=0}^{t-1} a^{t'}, & \text{if } t \geq 1 \\ \text{zero vector}, & \text{if } t = 0 \end{cases} \tag{8}$$

The coverage vector, an unnormalized probability distribution over the article tokens w_i, can be seen as history of the attention distribution up to decoding time step t. To allow the model to utilize the information captured by c^t, equation 3 is changed as follows:

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + w_c c_i^t + b_{\text{attn}}) \tag{9}$$

where w_c is a trainable parameter vector.

1.2.2 Training. See et al. [8] report best results when first training until convergence and only then train briefly train with the coverage mechanism. During training, teacher forcing is used: When trying to predict the t-th target token, the decoder is given the correct target token from step $t-1$.

The loss for a sequence of length T is

$$\text{loss} = -\frac{1}{T}\sum_t \log P(w_t^*) \tag{10}$$

The authors found it useful to change the the loss function as follows when using the coverage mechanism:

$$\text{loss} = -\frac{1}{T}\sum_t \left[\log P(w_t^*) + \lambda \sum_i \min(a_i^t, c_i^t)\right] \tag{11}$$

where λ is a hyperparameter. When attending to the same token w_i at multiple time steps, c_i^t will become large. If c_i^t is large, the model is better off generating a^t s.t. a_i^t is small in order to minimize the loss function. This way, the model model might learn to not attend to the same location multiple times. In turn, this may prevent the generation of the same token or phrase multiple times.

Triples	(‘Ampelkoalitionen’, ‘arbeiteten’, ‘Mönchengladbach und Remscheid’), (‘Bündnisse’, ‘bildeten’, ‘sich’)
Fact Description	‘Ampelkoalitionen arbeiteten Mönchengladbach und Remscheid || Bündnisse bildeten sich’
Sentence	Nach den Kommunalwahlen 2009 arbeiteten Ampelkoalitionen in den nordrhein-westfaĺĺlischen Grossstädten Bielefeld, Mönchengladbach und Remscheid, und nach den Kommunalwahlen in Nordrhein-Westfalen 2014 bildeten sich Bündnisse aus SPD, GRÜNE und FDP in der Landeshauptstadt Düsseldorf und in Oberhausen.’

Table 1: Example of an extracted fact description. We construct *(subject noun phrase, predicate, object noun phrase)* triples based on the dependency tree of the original sentence.

1.3 Auxiliary Fact Encoder

Cao et al. [4] observe that around 30% of generated summaries from state-of-the art abstractive text summarization systems suffer from fact fabrication. We found that fact fabrication is also an issue with our system. The authors address the problem by providing the model with *fact descriptions* in addition to the input article. A fact description is a sequence of tokens which attempts to capture the facts of an article; An example can be seen in Table 1. As illustrated in Figure 1, the fact descriptions are fed into an auxiliary bidirectional LSTM encoder.

Fact Description Extraction. Unlike in [4], we cannot use Stanford’s OpenIE [1] extraction tool to extract (subject, predicate, object) triples for German. On a per sentence basis, we use following approach instead. Refer to Table 1 for an example.

(1) Dependency parsing and named entity extraction on the sentence using spaCy.
(2) Extract tuples *(subject noun phrase, predicate)* and *(predicate, object noun phrase)* from the dependency graph.
(3) Merge tuples containing the same predicate to *(subject noun phrase, predicate, object noun phrase)* triples.
(4) Discard all triples that do not contain a named entity. We found that this step increases the subjective quality of the resulting fact descriptions.
(5) Concatenate all tokens of all triples in the order in which they appear in the article.

Integrating the Auxiliary Fact Encoder into the Main Model. As in [4], we provide the main model with the features extracted from fact descriptions by combining the attention context vectors of both encoders. To avoid more hyperparameters, we only use a linear projection followed by the sigmoid non-linearity rather than a multi-layer perceptron.

$$g_t = \sigma(W_f[h_t^{\text{article}}, h_t^{\text{fact}}] + b_f) \tag{12}$$

$$h_t^* = g_t \odot h_t^{\text{article}} + (1 - g_t) \odot h_t^{\text{fact}} \tag{13}$$

where $\odot$ denotes pointwise multiplication and W_f and b_f are learnable parameters. h_t^{article} corresponds to h_t^* in Equation 3.

1.4 Decoder

During inference, the previously generated output token is fed back into the decoder. In the first step, a special start-token <d> is supplied. As soon as the decoder generates the end-token </d>, the output is considered complete. We use beam search for decoding.

In this Section, we discuss three approaches to improve generated summaries.

(1) Suppression <UNK> tokens in the summary.
(2) Avoiding repeated tokens.
(3) Automatic summary length control.

<UNK> denotes the token that is used for all OOV tokens in the articles and summaries.

Suppressing <UNK>s. We discussed in Section 1.2 that the pointer-generator network is able to copy OOV tokens in the article to the summary. This is possible because these tokens are represented in the input by a temporary token-id from which the token can be recovered. Despite this mechanism, the generated summaries sometimes contain <UNK>s. Intuitively speaking, the network is unsure which token to copy from the article or to generate from to vocabulary distribution.

To overcome this issue, we set the probability of the <UNK> token in the final distribution to 0. Thus, in cases where the model would previously generate <UNK>, it is now forced to fall back on the token with the second highest probability.

Avoiding Repeated Tokens. Repeated tokens or phrases are a common problem when using sequence-to-sequence models. As discussed in Section 1.2, See et al. [8] introduce the coverage mechanism to address this problem. While we were able to reduce the the number of repeated noun phrases by around 50% in some preliminary experiments, the output sometimes still contains repeated tokens.

Rather than changing the architecture of the model or its training objective, we guide the beam search decoder towards hypotheses with few repeated tokens. Let $\boldsymbol{x}$ denote the sequence of tokens in the input article. $\boldsymbol{y}_i^t$ denotes the i-th hypothesis – i.e. a sequence of tokens – up to decoding step t.

Beam search expands its current set of hypotheses based on the conditional likelihood

$$\text{score} = \log p(w|\boldsymbol{x}, \boldsymbol{y}_i^t) \tag{14}$$

where w is a token in the extended vocabulary.

Based on [6], we modify this score to penalize hypotheses containing repeated tokens. Let $n = t + 1$ and n_{u} be the number of tokens and the number of unique tokens in $\boldsymbol{y}_i^t$, respectively.

$$\text{score}_{\text{u}} = \log p(w|\boldsymbol{x}, \boldsymbol{y}_i^t) + \alpha \frac{n_{\text{u}}}{n} \tag{15}$$

The second term, weighted by the hyperparameter α, captures unigram novelty.

Automatic Summary Length Control. During training, we condition the model on the number of tokens in the summary. We found that the the article length correlates with the summary length: The Pearson correlation coefficient is 0.531. Thus, during inference, we guide the model to generate a summary with appropriate length based on the article length.

Similar to [5], we group the training samples into eight similar sized bins according to the summary length and enumerate these bins in ascending order. For each training sample, we append the binary encoding of its corresponding bin label to the input-embedding of the decoder. Table 2 lists the bins that we used in our submission and their corresponding encodings.

Analogously, we group articles into the same number of similar sized bins. During inference we do not have a reference summary where we could get the target length from. Instead we append the binary encoding of the bin label – of the article bin into which the article falls – to the decoder embeddings.

1.5 Postprocessing

We use Mosestokenizer[2] do detokenize the generated summaries.

Furthermore, some words need to be capitalized. This includes entity names, nouns or the first token of a sentence. To do the latter we use NLTKs[3] sentence tokenizer to detect sentences. To capitalize nouns and entity names, we constructed a vocabulary of the 180′000 most frequent (cased) tokens from the unprocessed dataset. For each generated word, we check whether its capitalized version is more frequent than the lower case version. If so, the token in question is capitalized.

2 EXPERIMENTAL SETUP AND HYPERPARAMETERS

For our final submission, we train on 99′800 samples (99.8%) of the provided training set and do not use any additional data. During training, we computed the perplexity on the remaining 200 samples and found that our model converged after around 17 epochs. After that, as suggested in [8], we continue training for 3000 steps with the coverage mechanism ($\lambda = 1$). We use batches of size 16 and train with a learning rate of 0.16. Following hyperparameters were used for the architecture:

- *LSTM hidden state*: 320
- *Embedding dimension*: 300
- *Max. number of encoder states*: 400
- *Max. number of decoder states during training*: 50
- *Max. number of decoding steps during inference*: 120

For decoding, we set the weight of the unigram avoidance parameter in Equation 15 to $\alpha = 0.05$. Table 2 lists summary and article bins used for length conditioning.

3 RESULTS

Some example summaries are listed in Table 3. We find that most summaries are relevant to corresponding article and are in most cases grammatically correct. However, despite our automatic length control, the model tends to generate rather short summaries.

Label	Encoding	Summary bin	Size	Article bin	Size
0	000	(0, 16)	12125	(0, 283)	12414
1	001	(17, 21)	12032	(284, 372)	12522
2	010	(22, 26)	12943	(373, 469)	12489
3	011	(27, 31)	12577	(479, 583)	12572
4	100	(32, 37)	12026	(584, 729)	12537
5	101	(38, 46)	12312	(730, 930)	12490
6	110	(47, 63)	13136	(931, 1248)	12495
7	111	(64, ∞)	12849	(1249, ∞)	12522

Table 2: We group articles and summaries of the training set into eight similar sized bins. Summary and article bins have the same binary encoding. This allows us to automatically choose the target length of the summary during inference based on the article length.

1	Samuel Sullivan Cox war ein US-amerikanischer Politiker. Zwischen 1857 und 1865 vertrat er den Bundesstaat Ohio im US-Repräsentantenhaus.
2	Karl Georg Heinrich von Hoym war ein preussischer General.
3	Die neue Landeszentrale für politische Bildung in Niedersachsen ist eine körperschaft des Niedersächsischen Ministeriums für Wissenschaft und Kultur in Gelsenkirchen.

Table 3: Example summaries. Refer to the Appendix for the corresponding source articles

REFERENCES

[1] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging Linguistic Structure For Open Domain Information Extraction.. In *ACL (1)*. The Association for Computer Linguistics, 344–354. http://dblp.uni-trier.de/db/conf/acl/acl2015-1.html#AngeliPM15

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1409.0473

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[4] Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2017. Faithful to the Original: Fact Aware Neural Abstractive Summarization. *CoRR* abs/1711.04434 (2017). arXiv:1711.04434 http://arxiv.org/abs/1711.04434

[5] Angela Fan, David Grangier, and Michael Auli. 2018. Controllable Abstractive Summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*. Association for Computational Linguistics, Melbourne, Australia, 45–54. https://www.aclweb.org/anthology/W18-2706

[6] Lisa Fan, Dong Yu, and Lu Wang. 2018. Robust Neural Abstractive Summarization Systems and Evaluation against Adversarial Information. *CoRR* abs/1810.06065 (2018). arXiv:1810.06065 http://arxiv.org/abs/1810.06065

[7] Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage Embedding Models for Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, 955–960. https://doi.org/10.18653/v1/D16-1096

[8] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. *CoRR* abs/1704.04368 (2017). arXiv:1704.04368 http://arxiv.org/abs/1704.04368

[9] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Coverage-based Neural Machine Translation. *CoRR* abs/1601.04811 (2016). arXiv:1601.04811 http://arxiv.org/abs/1601.04811

[2] https://pypi.org/project/mosestokenizer/
[3] https://www.nltk.org/

A REFERENCE ARTICLE 1

Samuel Sullivan Cox wurde ungefähr neuneinhalb Jahre nach dem Ende des Britisch-Amerikanischen Krieges in Zanesville geboren und wuchs dort auf. Er besuchte die Ohio University in Athens und graduierte 1846 an der Brown University in Providence . Cox studierte Jura, erhielt seine Zulassung als Anwalt und begann dann 1849 in Zanesville zu praktizieren. Er erwarb die Zeitung "Columbus Statesman" in Ohio und war in den Jahren 1853 und 1854 dort als Redakteur tätig. 1855 ging er als "Secretary of the US Legation" nach Lima . Politisch gehörte er der Demokratischen Partei an. Er nahm als Delegierter in den Jahren 1864 und 1868 an den Democratic National Conventions teil. Bei den Kongresswahlen des Jahres 1856 wurde Cox im zwölften Wahlbezirk von Ohio in das US-Repräsentantenhaus in Washington, D.C. gewählt, wo er am 4. März 1857 die Nachfolge von Samuel Galloway antrat. Er wurde zwei Mal in Folge wiedergewählt. Dann kandidierte er im Jahr 1862 im siebten Wahlbezirk von Ohio für einen Kongresssitz. Nach einer erfolgreichen Wahl trat er am 4. März 1863 die Nachfolge von Richard Almgill Harrison an. Zwei Jahre später erlitt er bei seiner Wiederwahlkandidatur eine Niederlage und schied nach dem 3. März 1865 aus dem Kongress aus. Während der Zeit als Kongressabgeordneter hatte er den Vorsitz über das "Committee on Revolutionary Claims" . Cox zog am 4. März 1865 nach New York City, wo er wieder als Anwalt tätig war. Bei den Kongresswahlen des Jahres 1868 wurde er im sechsten Wahlbezirk von New York in das US-Repräsentantenhaus in Washington D.C. gewählt, wo er am 4. März 1869 die Nachfolge von Thomas E. Stewart antrat. Nach einer erfolgreichen Wiederwahl erlitt er im Jahr 1872 eine Niederlage und schied nach dem 3. März 1873 aus dem Kongress aus. Während dieser Wahl trat er sowohl für die Demokraten als auch für die "Liberal Republicans" für den "at-large"-Sitz im 43. Kongress an. Am 4. November 1873 wurde er dennoch in das US-Repräsentantenhaus gewählt, um dort die Vakanz zu füllen, die durch den Tod von James Brooks entstand. Er wurde fünf Mal in Folge wiedergewählt. Im Jahr 1884 kandidierte er im achten Wahlbezirk für einen Kongresssitz. Nach einer erfolgreichen Wahl trat er am 4. März 1885 die Nachfolge von John J. Adams an, verkündete aber am 20. Mai 1885 schon seinen Rücktritt. Als Kongressabgeordneter hatte er in dieser Zeit den Vorsitz über das "Committee on Banking and Currency" , das "Committee on the Census" , das "Committee on Foreign Affairs" und das "Committee on Naval Affairs" . Präsident Grover Cleveland ernannte ihn am 21. Mai 1885 als Nachfolger von Lew Wallace zum Gesandten im Osmanischen Reich âĹŠ eine Stellung, die er bis zum 22. Oktober 1886 innehatte. Am 2. November 1886 wurde er im neunten Wahlbezirk von New York in das US-Repräsentantenhaus gewählt, um dort die Vakanz zu füllen, die durch den Rücktritt von Joseph Pulitzer entstand. Cox wurde in die zwei folgenden Kongresse wiedergewählt. Er verstarb während seiner letzten Amtszeit am 10. September 1889 in New York City und wurde dann auf dem Green-Wood Cemetery in der damals noch eigenständigen Stadt Brooklyn beigesetzt. Sein Grossvater war der Kongressabgeordnete James Cox aus New Jersey. Er wurde nach Samuel Sullivan benannt, der zwischen 1820 und 1823 "State Treasurer" von Ohio war. Samuel Sullivan Cox war als redegewandter Sprecher bekannt. Seinen Spitznamen "Sunset" bekam er wegen einer besonders blumigen Beschreibung eines Sonnenuntergangs in einer seiner Reden James H. Baker, der damalige Redakteur der "Scioto Gezette", einer Whig-Zeitung in Chillicothe, gab ihm daraufhin den Spitznamen. Cox verfasste während seines Lebens die folgenden Werke:

B REFERENCE ARTICLE 2

Karl Georg Heinrich von Hoym wurde 1739 als Sohn von Hans Bogislaws von Hoym, Erbherr auf Poblotz, und dessen Frau Auguste Henriette, geborene von Wobeser, geboren. Sein Vater, damals preussischer Lieutenant, starb bereits 1741 im Ersten Schlesischen Krieg. Ein Jahr später verstarb auch die Mutter. Von Hoym wurde daraufhin von Heinrich Graf von Podewils aufgenommen und mit dessen Söhnen aufgezogen. Nach dem Besuch des Collegium Fridericianum in Königsberg begann von Hoym 1758 ein Jura-Studium an der Universität in Frankfurt an der Oder. Er verliert das Interesse am Studium und versucht mehrere Sprachen zu erlernen. Im Juli 1761 tritt er als Fahnenjunker in das Kürassierregiment von Gustav Albrecht von Schlabrendorf in Breslau ein. Von Schlabrendorf rät von Hoym allerdings "wegen seines schwächlichen Aussehens" zum Abschied und empfiehlt ihn seinem Bruder, dem dirigierenden Minister Ernst Wilhelm von Schlabrendorf. Von diesem wird von Hoym am 8. August 1761 als Auskultator an der Breslauer Kriegs- und Domänenkammer angestellt. Nach relativ kurzer Zeit wurde er am 29. April 1762 zum Kriegs- und Domänenrat ernannt. Im März 1767 wird er Geheimrat und zweiter Kammerdirektor. Im gleichen Jahr heiratete er Antonie Louise Freiin von Dyhern und Schönau . 1768 lernte ihn Friedrich der Grosse selbst kennen und ernannte ihn 1769 zum Regierungspräsidenten in Kleve und 1770 zum dirigierenden Minister in Schlesien, um welches sich Hoym sehr verdient machte. Friedrich Wilhelm II. verlieh ihm 1786 die Grafenwürde und betraute ihn 1793 auch noch mit der Verwaltung des neu erworbenen Südpreussen. Hier gab Hoym durch bürokratischen Despotismus sowie schlechte Verwaltung, Selbstbereicherung und Verschleuderung des Staatsguts grossen Anstoss und veranlasste so das Schwarze Buch von Hans von Held. Ab 1796 war er Inhaber der Dompropstei Kucklow in Hinterpommern. Ihr vorheriger Inhaber, der preussische Generalfeldmarschall Wichard von Möllendorff, hatte sie ihm mit Genehmigung des Königs übertragen. Nach dem Tilsiter Frieden wurde Hoym in den Ruhestand versetzt und starb am 22. Oktober 1807 auf seiner Besitzung in Dyhernfurt bei Breslau. Er heiratete 1767 die Freiin "Antonie Louise von Dyhern und Schönau" , eine Tochter des Hofmarschall und Kammerdirektors in Oels Freiherr "Anton Ulrich von Dyhrn" und der Freiin "Sophie Caroline von Crausen". Seine Frau war auch Erbin von Dyhernfurth. Das Paar hatte zwei Töchter.

C REFERENCE ARTICLE 3

Gegründet wurde die niedersächsische Landeszentrale für politische Bildung 1955 unter dem Namen "Niedersächsische Landeszentrale für Heimatdienst". Die Umbenennung in Landeszentrale für politische Bildung erfolgte 1959. Das öffentliche Interesse erregte 1966 die Meldung, dass dem ehemaligen SS-Mitglied und damaligen Mitglied des Niedersächsischen Landtages Otto Freiherr von Fircks durch Mittel der NLpB ein Besuch von Israel und der Gedenkstätte Yad Vashem ermöglicht wurde, ohne dass er die Landeszentrale für politische Bildung von seiner Vergangenheit in Kenntnis setzte. Zum 31. Dezember 2004 wurde die Landeszentrale von der niedersächsischen Landesregierung unter Führung des Ministerpräsidenten Christian Wulff und Uwe Schünemann aus Kostengründen aufgelöst. Dies führte zu erheblichen Protesten, unter anderem durch die Bundeszentrale für politische Bildung. Nach der Auflösung 2004 wurde die Arbeit der niedersächsischen Landeszentrale für politische Bildung von verschiedenen Organisationen übernommen: Im Jahr 2008 forderte die Fraktion Bündnis 90/Die Grünen in der 16. Legislaturperiode des Deutschen Bundestages in ihrem Antrag , dass die Bundesregierung auf die niedersächsische Landesregierung einwirken solle, damit wieder eine Landeszentrale für politische Bildung in Niedersachsen errichtet wird. Im April 2016 beschloss der Niedersächsische Landtag einstimmig die Wiedererrichtung einer Niedersächsischen Landeszentrale für politische Bildung. Sie wurde als nichtrechtsfähige Anstalt des öffentlichen Rechts im Geschäftsbereich des Niedersächsischen Ministeriums für Wissenschaft und Kultur errichtet und am 25. Januar 2017 eröffnet. Die Landeszentrale hat acht Mitarbeiter und ihr steht ein jährliches Budget von 870.000 Euro zur Verfügung. Der Sitz befindet sich im Zentrum von Hannover am Georgsplatz. Nach einstimmigen Votum des Kuratoriums der Landeszentrale, bestehend aus neun Angehörigen aus allen Fraktionen des Niedersächsischen Landtags, wurde Ulrika Engler als Direktorin bestimmt. Zuvor leitete sie seit 2007 die politische Bildungseinrichtung "aktuelles forum" in Gelsenkirchen. Nach der Auflösung der Landeszentrale im Jahr 2004 hatten staatliche und freie Träger die politische Bildungsarbeit übernommen, darunter Gedenkstätten, Gewerkschaften, Kirchen, Schulen, Stiftungen und Volkshochschulen. Diese Einrichtungen werden von der neu gegründeten Landeszentrale vernetzt und unterstützt. Die neue Landeszentrale tritt verstärkt im Internet in sozialen Netzwerken, wie Facebook, auf und verbreitet Filme auf Youtube. Damit soll Zugang zu politischen Informationen ermöglicht werden. Um die politische Ausgewogenheit der Arbeit der Niedersächsischen Landeszentrale für politische Bildung zu gewährleisten, wurde bis 2004 ein Kuratorium aus siebzehn Mitgliedern des niedersächsischen Landtages eingesetzt. Nach der Wiedergründung der Niedersächsischen Landeszentrale für politische Bildung gehören dem Kuratorium neun Personen an, die aus allen Fraktionen des Niedersächsischen Landtags stammen. Zum Vorsitzenden wurde Marco Brunotte gewählt.