And the Winner is: Huntington Nonalcoholic Antimalarial Brushless Delays

Steven Smits
s4232763
`Steven.Smits@student.ru.nl`

Tommy Clausner
s4836219
`t.clausner@student.ru.nl`

Abstract

Popularity in science is often measured by means of number of citations for a given set of scientific articles. Those metrices are used to compare the performance of scientists. Various research shows that the respective metric depends partly on the title of a given scientific article. It remains unclear which features highly cited articles have in common. For researchers in general, a classification system that can be used to sort out ineffective paper titles would be of great use.
The present work aims to find a solution to this problem by constructing a neuronal network architecture that is able to distinguish between titles of higher and lower quality in terms of the expected number of citations a given title would yield. A dataset of 577635 titles and corresponding number of citations was acquired and ASCII coded or coded using word embedding to represent titles in different numerical ways. Further, a convolutional neuronal network was trained to distinguish between high and low rank titles, yielding a final classification accuracy of $\approx 75\%$. Additionally, it was explored to what extent high rank titles can be generated using a generative adversarial network. The title of the present work was generated by one of our networks.

Contents

1 Introduction

Peer reviewed papers in scientific journals continues to be the main means of communication for the scientific community. They are highly important for scientists to keep up with the latest progress in their respective fields and to publish own experiments and findings. The latter dependency becomes apparent in that a researchers' publication metrics highly determine success in the academic job market (van Dijk et al. [2014]). Specifically, a researchers' number of publications, impact factor and citations are predictive of whom to choose for leading positions. Accordingly, universities primarily look at the list of publications and credit as a scientist during the selection procedure for hiring researchers. For a researchers' credit, the golden standard continues to be the amount of citations in peer reviewed journals (Neylon and Wu [2009]). It comes as no surprise that scientists experience a tremendous pressure to publish in order to demonstrate their academic competence (Rawat and Meena [2014]). With the overall numbers of publications vastly increasing, how could a scientist distinguish oneself in order to gain more credit? One obvious stratagem would be to somehow increase the number of citations per paper published. It is long debated that certain title features may be predictive of its success in terms of citations (Letchford et al. [2015], Jacques and Sebire [2010], Jamali and Nikzad [2011]). For example, shorter titles have been shown to boost citations, whereas titles containing question marks received fewer citations. With the goal of distinguishing oneself in the academic job market, it is clear that creating article titles with the potential of being extensively cited will aid this goal. The present project addresses this goal using two different approaches: 1. Check how differentiable highly cited and lower cited paper titles are, and 2. Generate a potential popular article title.

1.1 Classification of title popularity potential

The first approach of this project will be to provide a tool that classifies article titles based on their popularity. Classification requires the presumed classes to have divergent features between classes and shared features within classes. Given the aforementioned literature, this property is satisfied by the title-popularity problem (Letchford et al. [2015], Jacques and Sebire [2010], Jamali and Nikzad [2011]). That is, article titles that have many citations appear to have distinct features from article titles that are cited less.

Artificial neural networks (ANNs) have been shown to be useful as a method for classification problems in various fields (Bala and Kumar [2017], Schmidhuber [2015]). This computational model is inspired by the biological neural networks that comprise the human brain in that a multitude of interconnected neurons are modifying their connections in response to some input that is experienced (van Gerven [2017]). The basic unit of an ANN is the neuron, which receives the input and transmits output as a mathematical function of the input:

$$y = f(\sum_{i=1}^{N} w_i x_i)$$

Where x_i is the input neuron, w are the synaptic weights, y is the output and N the number of neurons. This model is often extended to contain multiple hidden layers of neurons a_j, before generating a final output y. In the classification problem, the output y will ideally be equal to target value t that is the label of the class to which x belongs. In reality, the true function that transforms the input to a target output is unknown. By updating the synaptic weights between neurons, the ANN multi-layer perceptron procedure can approximate any continuous function (Hornik [1991]). The main approach to find the optimal synaptic weights to estimate an unknown function is to minimize some cost function (Nielsen [2015]). One example cost function is the quadratic loss function:

$$E(w) = \frac{1}{2} \sum_{k} (y_k - t_k)^2$$

Where the cost E is zero if the predicted label y matches the target label t and the cost increases quadratically by any increase in predictive deviation. With an unknown true transformation function and increasing number of synaptic connections, the minima of this cost function cannot be found analytically. For this, techniques such as gradient descent are often used. Altogether, the network will learn to use underlying features belonging to classes in order to predict the correct class y based on input x.

The use of ANNs for classification is easily translatable to the problem of article title-popularity classification. To elucidate, target labels would be the popularity of an article ([no, yes]) and the input would be some numerical representation of the title. If the network is able to distinguish titles based on their popularity, one could input new titles to test whether those new titles would be classified as popular.

1.2 Generation of popular titles

The second goal is to generate a title that is a potential popular title in terms of how many citations a paper having this title would gain. Scientists often operate in highly specific subfields, which makes the generation of a random popular article title for their specific experiments senseless. The main aim of this part of the project is not to promote a generation of lazy scientists that create unrelated popular article titles. The two-fold aim is to: 1. Expose features exhibited by popular article titles by creating them, and 2. Making this project the most popular, by creating a popular title.

Title generation was realized using a Generative Adversarial Network (GAN). GANs are an upcoming technique to model high-dimensional distributions of data (Goodfellow et al. [2014, 2016], Creswell et al. [2017], Nguyen et al. [2017], Li et al. [2017]). The network is able to do this by simultaneously training two competing networks, hence the term *adversarial networks*. The discriminator network $\mathcal{D}$ tries to asses the probability that data is from some distribution $p(x)$, whereas the adversarial generative network $\mathcal{G}$ aims to generate data that tries to mimic coming from $p(x)$. The discriminator receives both samples from $p(x)$ and samples generated by $\mathcal{G}$, which it assesses the probability of the data belonging to $p(x)$. The discriminator can learn through its payoff $v(\theta^{(g)}, \theta^{(d)})$, which takes into account the labels of both the generated samples and the $p(x)$ samples. The generator has no access to the real distribution $p(x)$ and can only learn from feedback given by $\mathcal{D}$. The feedback $\mathcal{G}$ receives is its own payoff $-v(\theta^{(g)}, \theta^{(d)})$. Both $\mathcal{D}$ and $\mathcal{G}$ try to maximize their own payoff in a zero-sum mini-max game, such that convergence is at a minimum for both networks:

$$g^* = \arg\ \min_G \max_D v(G, D)$$

Where v is the cost function, that is usually characterized by some cross-entropy cost (Goodfellow et al. [2016]). The adversarial networks could be multi-layer perceptrons or more complex like convolutional networks. The standard algorithm typically uses a gradient descent procedure using minibatch training switching between $\mathcal{D}$ and $\mathcal{G}$ iterating over all the data for k steps. In an ideal situation, the algorithm stops when $\mathcal{D}$ only outputs $1/2$, meaning that $\mathcal{G}$ generates perfect samples from $p(x)$.

The use of GANs can be translated to the problem of generating popular article titles. To elaborate, the target distribution $p(x)$ can be the feature representation of popular article titles. The generator can then learn to generate some numerical representation that is close to samples drawn from $p(x)$. The discriminator can judge how close the generated samples are to the popular article titles. If both networks converged, the GAN could be used to generate random popular article titles.

1.3 Project aims

The aim of the project is to study whether highly cited academic articles have common features in their titles. Furthermore, it is investigated if neural networks can be used to tackle this question using a multitude of approaches in the field. First, it is aimed to discriminate highly cited titles from fewer cited titles. If discrimination is above chance level, this would suggest that titles do have features that are correlated with article popularity. Second, it is aimed to expose these feature using a GAN that learns to produce popular titles. It is expected that if popular titles have common features, these can be generated by a GAN.

2 Methods

2.1 Design

In order to use a neural network to approach article titles and citations, it is a minimum requirement to use a dataset that contains this information. This dataset should, at the very least, include titles and their respective citations. To the very best of the authors knowledge no such dataset is publicly available yet. For this reason the project started with engineering a Google Scholar crawler to acquire the respective dataset. After data acquisition, four neural networks were programmed to tackle the project aims. To clarify which approach of title representation is more suitable for the aforementioned problems, the data was represented in two types: 1. ASCII code, and 2. embedded words. The reason for this additional split was to inspect what level of data complexity is required for either classification or generation of popular articles titles. Separately for both ways of data representation a fitting classification network and a GAN were programmed. This yields four networks independent network architectures: 1. classifier ASCII code, 2. classifier word embedding, 3. GAN ASCII code, and 4. GAN word embedding. Figure 2 visualizes the respective difference in the analysis pipeline for ASCII and word embedded data representations.

2.2 Data acquisition and preprocessing

Data acquisition In order to acquire a dataset of sufficient size, the use of an autonomously working data mining device was inevitable. For this purpose a Raspberry Pi 2 B minicomputer (Pi) running Ubuntu Server 16.04 (ubuntu) was used. Since Google Scholar (Scholar) provides full access to the required data in an easy-to-use manner the respective database was chosen. One major downside of Google Scholar however, is the robot or web-crawler protection, that pops up an "I'm not a robot" text field and sometimes even small selection tasks that have to be made in order to proof being a human user. To overcome the mentioned limitations a time jitter between each new URL-request was used, which more than doubled the amount of data that could be obtained before eventually getting cought by Google. However, since the respective IP is blocked for several hours the final solution included further a continuous IP address renewal after the crawler was detected. By exploiting the fact that when logging into the University's VPN as an employee, the external IP address would change even after only several minutes, it was possible to speed up the process significantly.
Following the procedure of crawling and reconnecting, a data acquisition rate of ≈ 29000 titles and corresponding citations per day was achieved. Table 1 (left) gives a general overview of the properties of the datasets used for later analyses. Search phrases were generated after acquiring a few hundred titles using manual search phrases common in the field of neuroscience and AI (e.g. fMRI, neuronal network, artificial intelligence). Those titles were split into unique words and the resulting list was shuffled. A new search phrase was created by combining one random unique search item with the word "neuro", yielding a unique search phrase like "fMRI neuro" or "convolutional neuro". This way a few thousands of titles were acquired and the procedure was repeated one more time. The final search list contained 30259 unique words, that due to technical limitations, could not be used exhaustively. After 25 days the final dataset grew to 727872 titles and corresponding number of citations by using ca. 30% of the last search list. In order to keep the dataset reasonably small for home computing all additional acquired data will not be part of the analysis.
The crawling function that was used can be found at https://github.com/ckreibich/scholar.py. It was slightly modified in order to obtain data from multiple search pages, since the original version was only able to collect data from the first results page. A depiction of the data crawling architecture can be found in figure 1.

Data preprocessing As a first step the data was cleaned in the following way: all acquired data was merged and irregularities (e.g. empty lines, or titles that are not followed by a citation) were sorted out. Secondly all titles were excluded that contained more than 20 of 41 predefined special characters, that did not belong to a written language family that was derived from Latin or contained a "..." character (this was to ensure that improper represented titles that were cut during data acquisition are not part of the dataset). Due to low quality, 20.6% of the data had to be removed yielding a final size of the dataset of 577635 unique "valid" paper titles including citations.
Since this novel approach to evaluate scientific work is a relative young field amongst AI, a preferred model for representing paper titles in numerical values is lacking. In fact there are two possible options: To encode the raw title into a string of characters that translate into a numerical system using e.g. the ASCII code (precisely: ASCII decimal). A second option would be to phrase the task as a form of sentence classification and use word embedding, that is relating all words across all titles based on the relative proximity between those words Bengio et al. [2006]. For the latter one, a dictionary containing the 20200 most occurring words over all titles was created. All words that occurred fewer times were replaced with the string "<unknown>". This was done to ensure a common

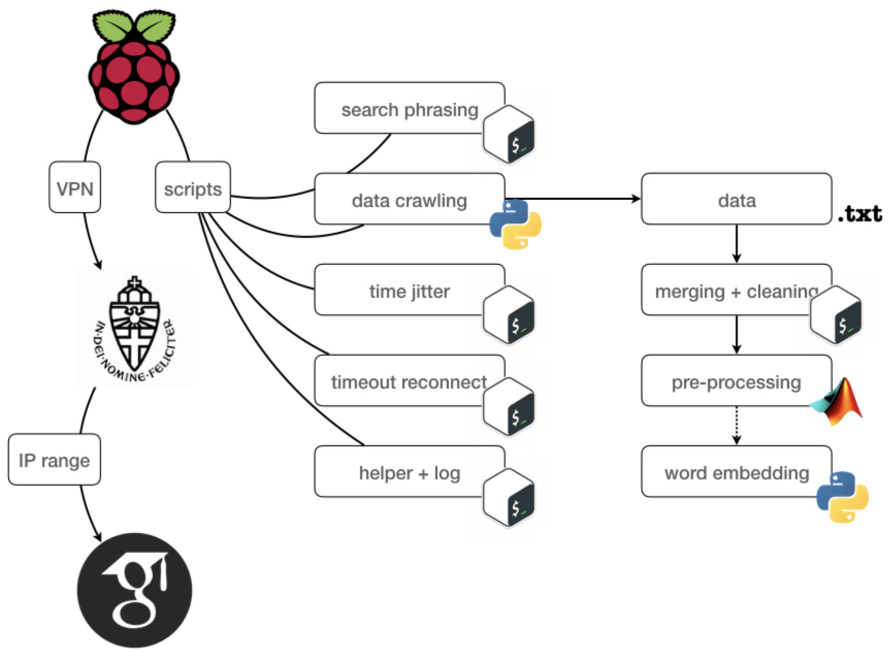

Figure 1: General data acquisition architecture and preprocessing pipeline. The actual data mining was preformed on a Raspberry Pi 2 B running Ubuntu Server 16.04. Via VPN the crawler device would connect to the University's network. Almost every time this happens a new external IP address will be assigned to the Raspberry PI. Exploiting this fact a controller script hands over a search phrase to the actual crawler, that searches a given URL for respective embedded fields (in the present case title and citation information). Using a time jitter to switch between search phrases and search pages allows for an extented crawling time before Google's robot detection is triggered, which in turn would cause a reconnect to the University's VPN to obtain a new IP address. Resulting text files included title and citation information, which where handed over to the preprocessing pipeline. Note that the respective icon at each subsection of the architecture indicates in which programming language the respective step was performed (cube: bash; two iconic snakes: Python; 3D plot: MATLAB). Corporate logos on the left represent the Raspberry Pi minicomputer (iconic Raspberry), Radboud University Nijmegen (detailed logo in the middle) and Google Scholar (iconic g).

language base for all titles. "<unknown>" words thereby were most often detailed, field specific specialist terms. Thus "<unknown >" in most cases represents field specific scientific language, which makes them more predictable. End of lines statements (i.e. end of titles) were marked using "<eol>". Afterwards all titles were translated into indices. Each index represents the position at which the respective word could be found in the dictionary. Afterwards all titles were concatenated to one vector of word indices.
Using the Skip-Gram algorithm, a neural network using only a single layer of n units is trained to learn relationships between words in a sentence. This way the respective weights serve as the respective word representation of n dimensions. Skip-Gram uses a window of a given size and slides over the vector, obtaining statistics of which words co-occur in which context. In the presented case a window size of 5 was used, meaning that given a word w_i all words between and including w_{i-5} and w_{i+5} are taken into account. Further, words were embedded using 200 dimensions. The network was trained for 20 epochs. A more detailed explanation can be found at https://docs.chainer.org/en/latest/tutorial/word2vec.html. Further, word embedding was performed using the following script that can be obtained at https://github.com/chainer/chainer/tree/master/examples/word2vec. The final result of this procedure

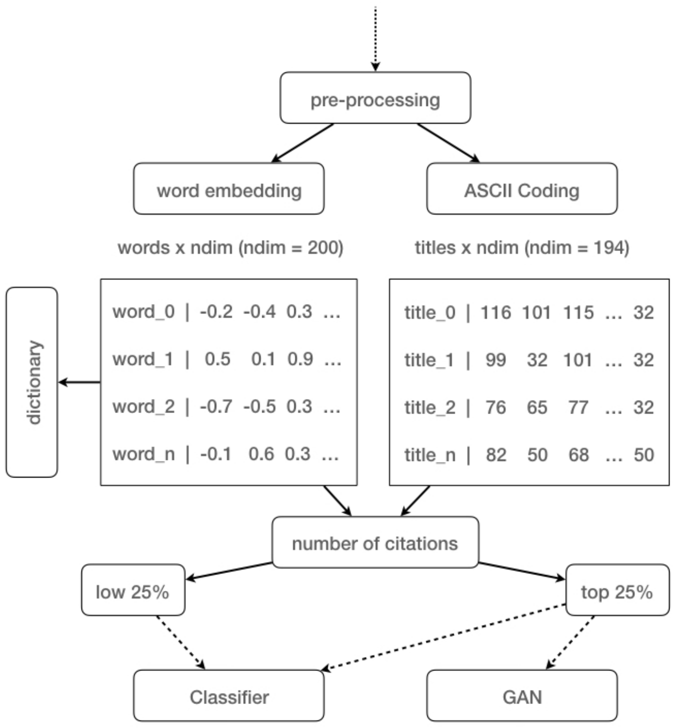
pre-processing
word embedding
ASCII Coding
words x ndim (ndim = 200)
titles x ndim (ndim = 194)
dictionary
word_0 | -0.2 -0.4 0.3 ...
word_1 | 0.5 0.1 0.9 ...
word_2 | -0.7 -0.5 0.3 ...
word_n | -0.1 0.6 0.3 ...
title_0 | 116 101 115 ... 32
title_1 | 99 32 101 ... 32
title_2 | 76 65 77 ... 32
title_n | 82 50 68 ... 50
number of citations
low 25%
top 25%
Classifier
GAN

Figure 2: General pipeline for data processing, comparing ASCII coded titles vs. word embedded titles. After cleaning the data, including the removal of incomplete datasets, excluding titles that expose more than 20 special characters or ". . ." at the beginning or end of the titles, further processing was split into two parts: ASCII coding and word embedding. ASCII coded titles were directly translated into ASCII decimal representation padding shorter titles with the value 32, which represents white space. Word embedding was done using the Skip Gram algorithm, that relates co-occurring words. Further a dictionary was created containing the 20200 most occurring words. All words of fewer occurrence where replaced with "<unknown>". End of line was coded using "<eol>". From both sets two subsets were extracted based on whether the respective number of citations was below the 25th percentile (low 25%) or above the 75th percentile (top 25%). The number of dimensions (ndim) refers to the number of columns the resulting matrix would have. In the ASCII coding ndim equals the maximum number of characters found in the longest title, whereas in the word embedding case ndim refers to the size of the vector representing each word.

is a matrix W of size $N_{words} \times 200$.
In turn, the ASCII coding was performed by obtaining the length of the title with the most characters, padding all other titles with *space* to reach this respective length and transforming the resulting character matrix of $N_{titles} \times ndim_{max}$ into ASCII decimal values. A more comprehensible graphical illustration can be found in Figure 2.
To distinguish between high and low rank titles, titles exposing a number of citations of above the 75th or below the 25th percentile of the distribution of all numbers of citation were selected. Basic statistical information about the distribution of citations can be found in Table 1 (right).

2.3 Title popularity classification

To classify titles based on popularity, the dataset was split into two sets to classify: popular and unpopular titles. The popular titles were defined as the upper 25% cited titles and the unpopular titles were defined as the lower bound 25% cited articles.

File stats			Citation stats		
Which	Unit	Value	Average	Median	Standard deviation
Full dataset	titles	727872	-	-	-
Cleaned dataset	titles	577635	447.9	143	1612.4
Upper 25%	titles	144517	1454.9	807	2999.7
Lower 25%	titles	144846	5.6	3	6.7
Dictionary	words	20200	-	-	-
Search list	words [+ neuro]	$\approx$ 10000	-	-	-
Crawl time	days	$\approx$ 25	-	-	-

Table 1: General descriptives of the dataset used. (left) Descriptives on the raw data. Respective values indicate basic information mainly on the size of the respective dataset. (right) Overview of basic statistical features the used data exposed, that is average, median and standard deviation of the number of citations for the respective groups.

ASCII code A neural network was used for binary classification of popularity of titles. Classification using ASCII code as data representation was accomplished by labeling unpopular targets as 0 and popular targets as 1. The input was an array of numbers that represented titles translated to ASCII code of length 194. The training subset represented 90% of the original data and the testing subset the remaining 10%. The neural network was a multi-layer perceptron consisting of 13 layers that alternated in activation functions used. The first hidden layer was defined as $h1 = ReLu(\sum_{i=1}^{N} w_{i1}x_i)$ with $ReLu$ being the rectified linear unit activation function. The second hidden layer was defined as $h2 = tanh(\sum_{i=1}^{N} w_{i2}h1_i)$ with $tanh$ being the hyperbolic tangent activation function. The last $tanh$ hidden layer was followed by a final layer $y = softmax(\sum_{i=1}^{N} w_{i12}h12_i)$ with the $softmax$ being the normalized exponential function that is used as a generalization of logistic function to classify the two-dimensional choice as a probability vector. The hidden layers had hidden units ranging from 25 to 1000, with the larger numbers at the lower layers. The network was optimized using adaptive gradient descent with a learning rate of 0.001. The softmax cross entropy function was used to calculate loss. The network was run for 100 epochs and training/testing accuracies (correctly classified titles) were calculated as mean accuracy per epoch.

Word embedding Performing the classification of the word embedded data was done similarly to the ASCII coded classification: A binary decision whether a title belongs to the highly cited collection of paper titles or to the respective lower end, was intended. Therefore accordingly target values were set to 0 (less-cited paper titles) or 1 (highly cited paper titles). As input data a matrix of size $batchsize \times 1 \times max_{words} \times ndim_{embedding}$ was used where the $batchsize$ was set to 72 titles ($80 - 0.1 \cdot 80$ random exclusion). max_{words} refers to how many words the longest title in the dataset contained (49 in the present case). All titles exposing less words were padded with vectors of zeros, that were coded as white space in the dictionary. The dimension $ndim = 200$ was used for the word embedding described earlier.
Since literature about sentence classification suggests the use of convolutional neuronal networks (Kalchbrenner et al. [2014], Kim [2014]), the following architecture was used: A fully connected input layer using a rectified liner activation function ($h = ReLu(\sum_{i=1}^{N} w_{i1}x_i)$) passed its activation to a 2D convolutional layer. This layer was set up to have one in- and one output channel and a kernel size of 3. Subsequently, a rectified liner activation function was used after which max pooling was applied using a kernel size of 3. This result served as input for the following two linear layers. The first was used to reduce dimensionality to 50, followed by the respective output layer having

a dimensionality of 2. The former was again activated by a ReLu activation function, followed by 20% dropout. A sigmoid activation function ($f(x) = (1 + \exp(-x))^{-1}$) was used for the respective output layer to map values between 0 and 1. Adaptive gradient decent was used to optimize the network. A learning rate of $lr = 0.001$ was chosen. To calculate the loss, softmax cross entropy between target and predictions was calculated.
The network was trained on 143200 example titles from both popular and unpopular targets, yielding 286400 titles. 800 examples of both sets were used as validation that was randomly chosen beforehand and which was never used to update the classifier. A full sampling through the whole dataset was done for 5 epochs, yielding 8950 training iterations.

2.4 Popular title generation

ASCII code A GAN was implemented with the aim to generate popular article titles. The data were the top 25% cited articles translated to ASCII code of length 194. In contrast to the classification, the titles were now normalized to range between 0 and 1. The algorithm structure was such that it could run for X iteration batches, with the discriminator updating first and the generator updating second each iteration. Both networks were optimized using stochastic gradient descent with a learning rate of 0.001 and a momentum of 0.9. The network used a batch size of 10, ran for 100 epochs (100000 iterations) and loss/accuracies were calculated as a mean per epoch.
For the generator, the input was uniformly Gaussian distributed noise of length 194 ranging between 0 and 1. The generator had an output size of 194 to represent titles in the length used for the discriminator. The neural network was a multi-layer perceptron consisting of 10 layers that alternated in activation function used. The first hidden layer hidden layer was defined as $h1 = ReLu(\sum_{i=1}^{N} w_{i1}x_i)$ with $ReLu$ being the rectified linear unit activation function. The second hidden layer was defined as $h2 = \sigma(\sum_{i=1}^{N} w_{i2}h1_i)$ with σ being the sigmoid activation function. Because there were 10 layers, the last layer was a sigmoid activation function that outputs values ranging between 0 and 1. The hidden layers had units ranging from layer 1 having 194×10 units, with the latter multiplier reducing by 1 per layer, such that the output layer is 194×1 units. The cost function was implemented as a least squared function as implemented by Mao et al. [2017]: $L_G = \frac{1}{2}(D(G(z) - 1)^2$, with z being the noise input to the generator.

Next, the discriminator had to distinguish generated popular titles from real popular titles. The output must be a scalar that represent the inputs' probability of being a real popular article title. The discriminator must distinguish two inputs in the sense that it has to output the probability that the input x belongs to the real data $p_{data}(x)$. The first input were samples from the real data set, which were the real popular titles represented as normalized ASCII code values ranging between 0 and 1. The second input were samples generated by the generator $G(z)$. The neural network structure of the discriminator was the same as the one used in the classification except that the last layer was now a sigmoid function to accomplish a scalar output probability value. The loss function was the one implemented by Mao et al. [2017], that penalizes data that is correctly classified but still far from the real data. This method has been shown to overcome the vanishing gradient problem. The loss function is defined as: $L_D = \frac{1}{2}(D(x) - 1)^2 + \frac{1}{2}D(G(z))^2$, with x being samples from the real data (popular titles). Finally, a dropout of 0.4 was used for the real samples input to the discriminator as well as label noise of 0.5 (i.e. randomly swap labels of real and fake samples, Salimans et al. [2016]).

Word embedding In order to generate highly popular titles from the word embedded dataset, a generative adversarial network (GAN) was used. Thereby the discriminator would learn to distinguish if the presented data came from the original set of the upper 25% most cited papers or if they were generated by the generator. The generator in turn would learn which features make a title belonging to the aforementioned group and tries to mimic them.
For the discriminator a convolutional neuronal network was used. A convolutional layer with 1 in- and 1 output channel and a kernel size of 3 would be activated using the ReLu activation function ($h = ReLu(\sum_{i=1}^{N} w_{i1}x_i)$), followed by a max pooling using a kernel size of 3. The respective output was passed to the output layer by applying a dropout of 20%. The output layer reduced the dimensionality to 2. A sigmoid activation function ($f(x) = (1 + \exp(-x))^{-1}$) was used to map the output between 0 and 1.
In turn, the generator was realized as a deconvolutional network. A fully connected ReLu input layer passes its output to the deconvolutional layer. This layer was set up to have 1 in- and 1 output

channel and a kernel size of 3. It was activated using a ReLu function and passed its output to another fully connected layer. As input data, the generator received randomly generated uniformly distributed values using the respective minimum and maximum of the original word embedding matrix as margins for the random value distribution. Generated title matrices were transformed back to human readable titles using the minimum $L2$ norm between the generated title vectors and each row of W. The row index of the smallest distance between w_{gen} and W was used to extract the respective human readable word from the dictionary.
Both networks were trained using stochastic gradient decent employing momentum. Momentum was set to 0.9, whereas the learning rate was set to 0.01 for the generator and 0.001 for the discriminator. The loss was calculated using softmax cross entropy between predicted and targeted class label. Note that the generator tries to optimize towards being labeled as class 1, whereas the discriminator tries to label data generated by the generator as 0. A batchsize of 8 titles ($10 - 2$ randomly excluded) was used and thus the network was trained on 115200 randomly selected and generated titles from the set of high ranking papers. Due to hardware limitations only one epoch was used for training, yielding 14400 training iterations. Examples for generated titles can be found in the appendix.

3 Results

3.1 Title popularity classification

ASCII code The accuracies for both the training and validation set that were calculated per epoch are depicted in Figure 3. The results show that the accuracy on the training set is improving over epochs, to 85% after 100 epochs. This implies that the classifier is able to distinguish known popular from unpopular articles based on their title. In contrast, the results also show that classification accuracy on the validation set doesn't improves and stays around 55%. Given that the classification problem is binary, the classifier's performance on the validation dataset seems to be around chance level. This divergence in training and validation accuracies suggests that the classifier network is over fitting. Moreover, the results suggest that the current implemented classifier in combination with the ASCII code data representation is unable to predict popularity potential of unknown article titles.

Word embedding In case of the classification of word embedded titles accuracy was computed after each iteration (training data) or after every 25 iterations (validation data). Figure 4 depicts classification accuracy and loss for the validation dataset (top) and for the test dataset (bottom). The validation dataset converged at an accuracy of $\approx 76\%$. Training accuracy increased continuously within the first epoch and stepped upwards after the onset of a new training epoch, peaking at $\approx 95\%$. Note, that the stepwise increase in accuracy for the training data clearly reflects a symptom for overfitting.

3.2 Popular title generation

ASCII code The loss for both the discriminator and the generator and the accuracy of the discriminator are depicted in Figure 5. The generator starts with a low loss in the first few epochs upon which it is seemingly converged to a maximum loss. Around 40 and 80 epochs the generator loss is vastly reduced. In these spikes the generator seems to be able to fool the discriminator, which is supported by a 50% accuracy of the discriminator at those points. The discriminator converged fast to an accuracy of 98%. This implies that the generator is not able to fool the discriminator, which could mean the discriminator is too powerful or the generator is too weak. Moreover, in combination with the classification results, the high accuracy suggests that the discriminator is overfitting to the real samples. Unfortunately, several overfitting techniques were applied and networks of varying strength were tested and most of them had the same results. Further, looking at the generators generated samples over epochs an interesting pattern emerged. The generator was able to generate an array of 194 in ASCII code that had values between 97 and 122 for the first 36 indices and near 0's for the subsequent indices. Translating this to characters, this corresponds to random letters for the former and empty space for the latter. Thus, the generator seemed to be able to create some feature of the original distribution of real titles (i.e. characters at beginning, embedding at end of representation vector). However, the results show that the implemented GAN and the used word representation is unable to produce sensible titles, let alone popular titles.

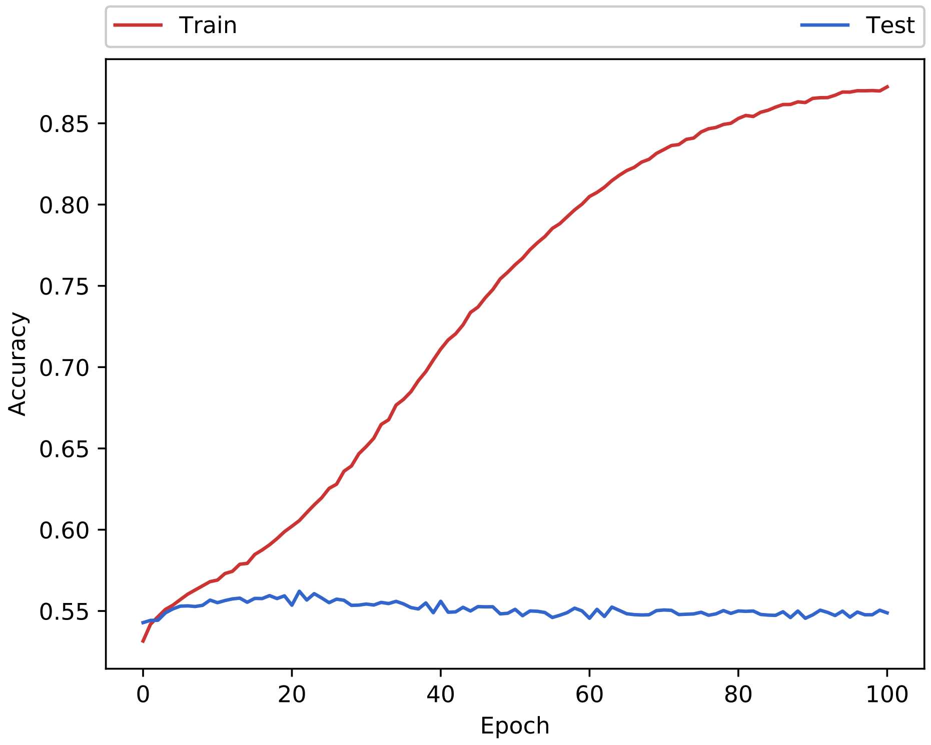

Figure 3: Classification results in ASCII code. The red line represents the classification accuracy for the training dataset, whereas the blue line represents the classification accuracy for the validation set.

Word embedding Discrimination accuracy and loss for the discriminator as well as for the generator can be found in Figure 6. The discriminator learns to differentiate between real and artificial samples after a few iterations, reflected by a steep increase in classification accuracy, ceiling at 1. Further the loss decreases in the very same way, reflecting the respective learning process. As for the generator, the loss is constantly increasing, reflecting the fact, that the generator was not able to generate numerical patterns that the discriminator would classify as coming from the original dataset.

4 Discussion

The aim of this project was to examine whether highly cited academic articles have common features in their titles. This was approached by classifying and generating titles based on their popularity as reflected by citations. Supplementary, it was examined which level of complexity the data representation needs to be in order to find such potential features. This was realized by classifying and generating titles using both ASCII decimal code and word embedding as data representations. It was hypothesized that if highly cited or scarcely cited titles have features in common, a neural network classifier could distinguish the popularity of those articles using only the title information. The results show that only when using word embedding as data representation, a neural network can distinguish popular from unpopular academic articles with an accuracy of $\approx 75\%$. Therewith, it was hypothesized that if classification is possible, features of popular titles could be exposed using a GAN. The results show that the implemented GANs were seemingly not able to generate sensible titles that express common features. Unifying all findings suggest that popular articles and unpopular articles have distinct features in their titles that allows one to predict their class. It remains to be revealed what those features are.
Previous research has shown that one feature of highly cited articles is having a short title (Letchford et al. [2015]). This is partly in line with this studies' generated titles in ASCII code. In particular, character density is inversely related to the index of the title vector. Thus, popular generated titles have more characters in the beginning of a fixed-length sentence than the end. The generated titles,

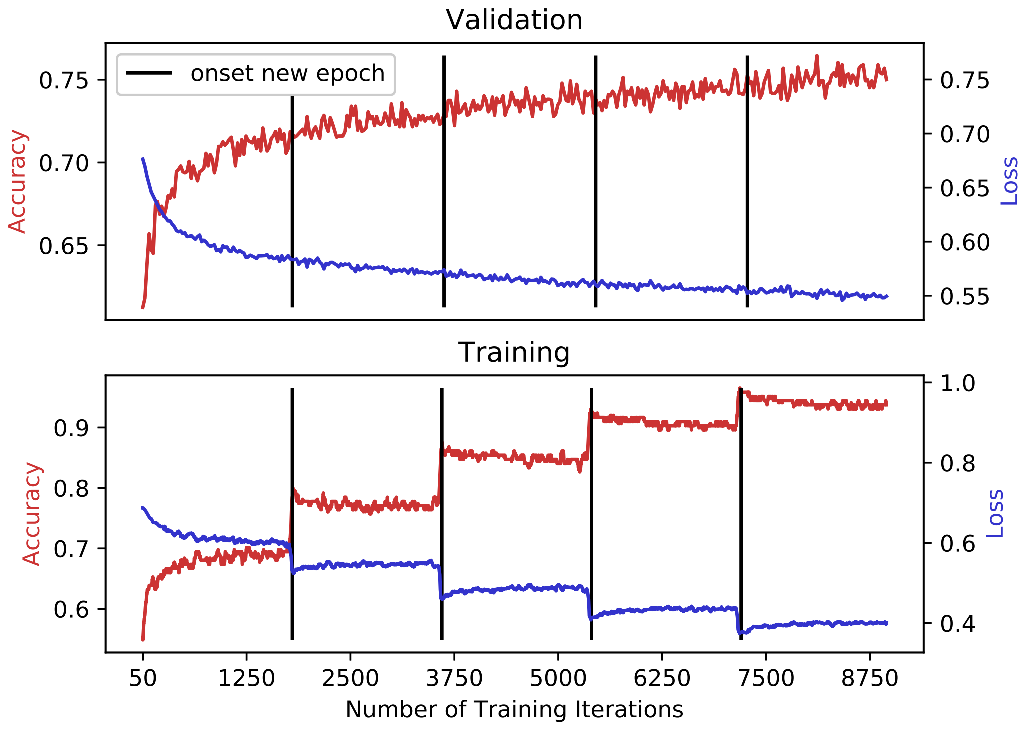

Figure 4: Title classification results based on embedded titles. Blue lines indicate the loss and red lines the accuracy for the respective test dataset over training iterations. (top) for the validation dataset and (bottom) for the training dataset. Black lines indicate the onset of a new training epoch. Note that between black lines the full dataset was used. However due to random exclusion of 10% of the data for each iteration, the training data between each epoch varies $\approx 10\%$.

however, are far from actual words and this outcome could simply be the result of padding the titles to equal length. Research has also shown that highly cited titles have fewer odd characters, such as "?", in their title than less cited articles (Jamali and Nikzad [2011], Jacques and Sebire [2010]). Confirmatively, the word embedded GAN did barely produce any odd characters even though they were part of the used dictionary. Thus, the results of this project are mostly in line with previous research on popular title features.

The ability of the classifier using word embedding as data representation to distinguish popular from unpopular titles with a reasonably high accuracy of $\approx 75\%$ is notable. It is tempting to conclude that to being highly cited in academic journals solely relies on good title craftsmanship. However, this study was not experimental and many other conclusions could be implied. It is likely that scientist that write high quality articles also formulate high impact titles. This would mean that titles do not affect citation rate and their relationship is only correlational. Notwithstanding this, it is staggering that article popularity can be distinguished at a reasonably high accuracy solely on their titles.

As for the results of the classification of word embedded titles, it is worth pointing out that after the full dataset was fed through the network, the majority of important features for classification were already picked up. Each of the additional training epochs was of limited benefit for validation accuracy. Furthermore, as clearly apparent in Figure 4 (bottom), the training network started to overfit more and more with the onset of each epoch. This phenomenon might explain the stepwise increase of training accuracy at the beginning of each new training epoch. However, the network was able to extract at least some additional generalizable information, as can be seen in Figure 4 (top) as there is still a continuous increase in validation accuracy in the validation set. In terms of a computational time vs. output trade-off it might be advisable to only train on the full dataset once, since the validation accuracy already reached a level close to its later maximum.

Since the convolutional network was able to extract popularity related title features, but neither of the

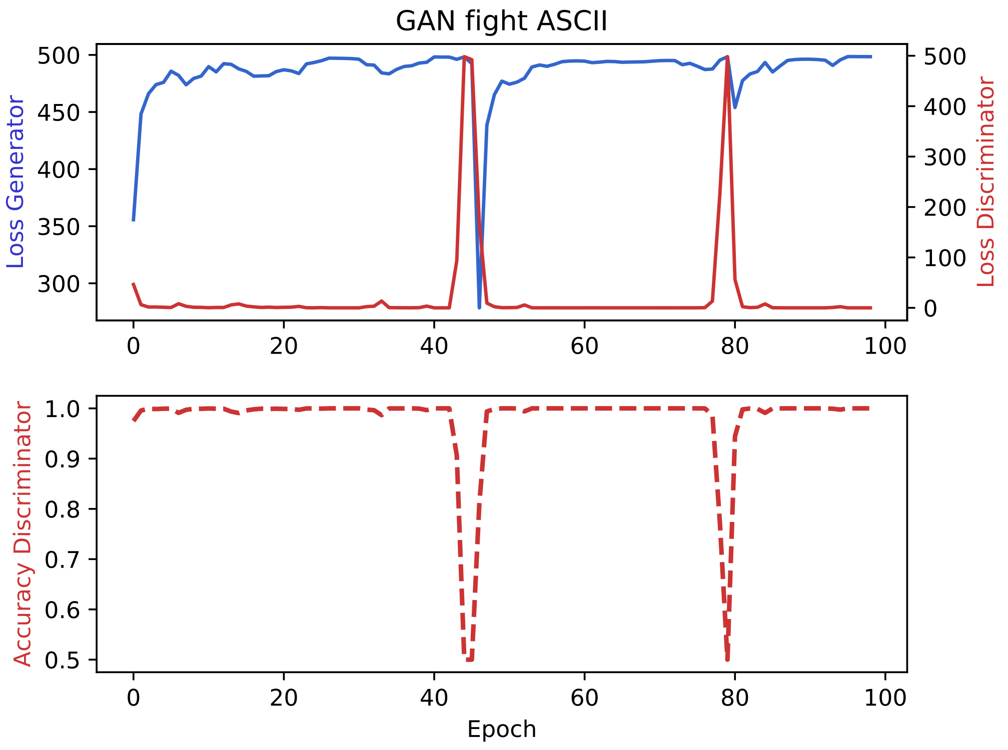

Figure 5: Results for the generative adversarial network (GAN) for title generation based on ASCII coding. (top) loss for generator (blue) and discriminator (red) over training epochs. (bottom) discrimination accuracy of the discriminator over training epochs.

two proposed GANs were able to generate sensible titles, a few limitations to this study need to be pointed out.
Computational power turned out to be the most limiting factor. Since only home computing devices such as private laptops were used to perform the respective analyses, computation time was a hurdle. The use of high-performance GPUs would have allowed for more training iterations and a way larger dataset. Due to a lack of computing facilities all computations had to be kept below $\approx 20h$. As reported by Ryoo et al. (2008) a performance increase of up to factor 450 can be achieved using GPU computing on high-performance GPUs as compared to CPUs.
The computational performance limitation directly links to a second issue, that is the word embedding itself. Due to limitations on hardware performance only 20 training epochs were used during the word embedding. The respective loss during the process clearly decreased, however it was far from convergence. Given the complexity and diversity of scientific paper titles as compared to a natural language sentence classification task, a more elaborate embedding might be inevitable. Future research therefore might focus on especially this issue. Finding and appropriate solution for scientific title embedding in its inherent sentence complexity remains a challenging task for network architects and hardware likewise. Previous research on word embedding suggests that increasing the dimensionality of the representation increases its embedding accuracy Levy and Goldberg [2014]. Thus, increasing the dimensionality of the word vectors from 200 up to 1000 might be desirable. Furthermore, running the word embedding for as many epochs as necessary to converge, would likewise increase the value of the data used for later classification or generation of paper titles.
Additionally, one might come up with a more appropriate solution for "<unknown>" words. Since scientific language is very specific in its nature, it would be rather advisable to either limit the data to a specific field (e.g. AI) to avoid the occurrence of "<unknown>" words in general. On the other hand the representation "<unknown>" can be seen as an indicator for subject specific knowledge, that then in turn would serve as a feature for generalization.
Additional improvements might also be achievable by using different network architectures. Deeper

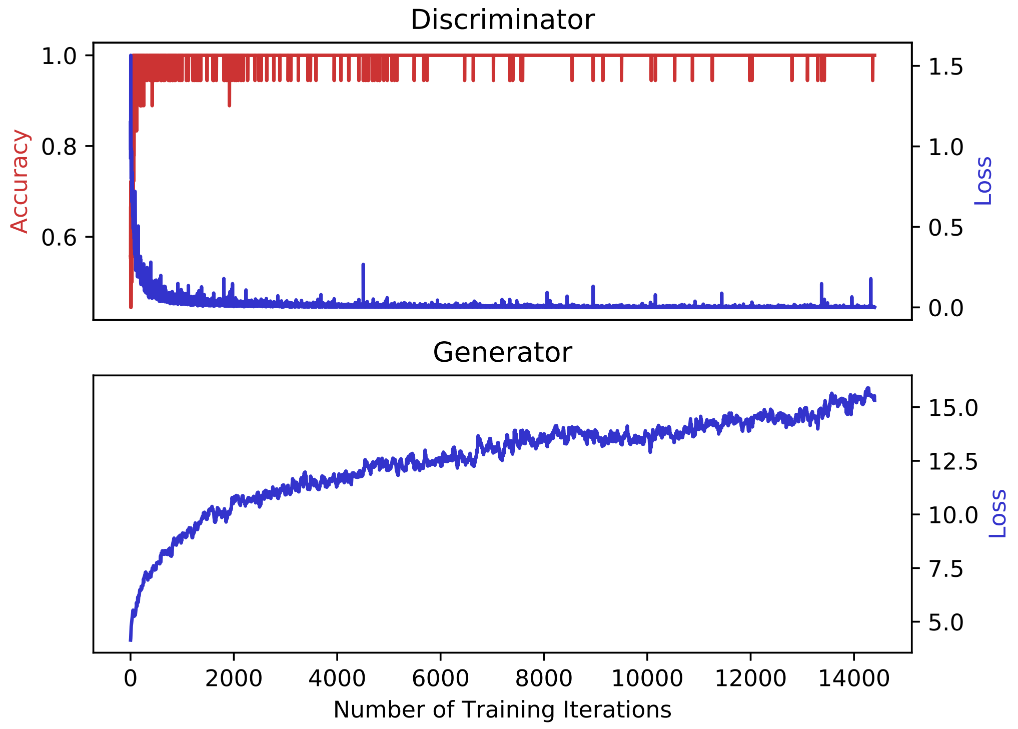

Figure 6: Results for the generative adversarial network (GAN) for title generation based on word vectors. Blue lines indicate the loss and red lines the accuracy for the respective network over training iterations. (top) for the discriminator and (bottom) for the generator.

and broader networks might be more easily able to pick up the complexity of the input data. A sensible way of feature reduction would probably increase performance even further.
In general more researcher has to be conducted in order to solve the problem of scientific title classification and generation. To this point it is unclear whether the convolutional network classifier picked up "good" features of high rank titles or rather was able to make the differentiation based on "bad" features of lower rank titles. Since this fact remains unclear, high rank title generation could have failed simply because the dataset used did not contain any distinct features. To clarify this issue, a generative approach would need to be used on the low rank titles as well as to all titles in general. Future research therefore has to improve on the problem of title representation, classification and generation. For title representation, mainly the word embedding procedure needs to be improved, which could be solved by increasing hardware performance. For the second issue of increased title popularity classification, many approaches might lead to Rome. For example, titles could be classified not only by splitting the dataset into two groups, but by directly predicting the number of citations a given title would yield. As error function the root mean square error (RMSE) could be used to define the loss between prediction and actual number of citations. Furthermore, number of citations should be transformed from an absolute number to citations per year. This way effects of scientific language that change over years, as well as sheer time effects will be excluded (note that the longer a paper is available the more likely is it to find the paper in a higher number of citation category). However the time factor might be only of minor importance, since the high rank group centered around 1455 citations (mean) or 807 (median) per article. These numbers can be considered "high" in general, even letting the factor time at the side. One might also want to normalize citations by impact factor of the journal, since the journal impact factor highly determines citations received (van Dijk et al. [2014]).

Conclusion The project sought to investigate whether popular academic articles can be distinguished from unpopular academic articles based solely on their title. It was found that a neural network could be trained to classify article popularity based on their title with high accuracy.This suggests that either popular articles or unpopular articles have common title features. Future research could shed light on what these features are.

References

R. Bala and D. Kumar. Classification Using ANN: A Review. *International Journal of Computational Intelligence Research ISSN*, 13(7):973–1873, 2017.

Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. Neural Probabilistic Language Models. In D. E. Holmes and L. C. Jain, editors, *Innovations in Machine Learning*, volume 194, pages 137–186. Springer-Verlag, Berlin/Heidelberg, 2006. ISBN 978-3-540-30609-2. doi: 10.1007/3-540-33486-6_6.

A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath. Generative Adversarial Networks: An Overview. *arXiv preprint arXiv:1710.07035*, 2017.

I. Goodfellow, Y. Bengio, and A. Courville. Generative Adversarial Networks. In *Deep Learning*. MIT Press, 2016.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*, pages 2672–2680. 2014. ISBN 1406.2661.

K. Hornik. Approximation Capabilities of Muitilayer Feedforward Networks. *Neural Networks*, 4: 251–257, 1991.

T. S. Jacques and N. J. Sebire. The impact of article titles on citation hits: An analysis of general and specialist medical journals. *JRSM short reports*, 1(1):1–5, 2010.

H. R. Jamali and M. Nikzad. Article title type and its relation with the number of downloads and citations. *Scientometrics*, 88(2):653–661, 2011.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

A. Letchford, H. S. Moat, and T. Preis. The advantage of short paper titles. *Open Science*, 2(8): 150266, 2015.

O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

C. Li, K. Xu, J. Zhu, and B. Zhang. Triple Generative Adversarial Nets. *arXiv preprint arXiv:1703.02291*, 2017.

X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley. Least Squares Generative Adversarial Networks. 2017.

C. Neylon and S. Wu. Level metrics and the evolution of scientific impact. *PLoS biology*, 7(11): e1000242, 2009.

T. D. Nguyen, T. Le, H. Vu, and D. Phung. Dual Discriminator Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2667–2677. 2017.

M. A. Nielsen. Neural Networks and Deep Learning. 2015.

R. Pi. Raspberry Pi - Teach, Learn, and Make with Raspberry Pi. `https://www.raspberrypi.org/`.

S. Rawat and S. Meena. Publish or perish: Where are we heading? *Journal of research in medical sciences: the official journal of Isfahan University of Medical Sciences*, 19(2):87, 2014.

S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, and W.-m. W. Hwu. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 73–82. ACM, 2008.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. 2016.

J. Schmidhuber. Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61:85–117, 2015. ISSN 18792782. doi: 10.1016/j.neunet.2014.09.003.

G. Scholar. Google Scholar. `https://scholar.google.com/`.

ubuntu. Ubuntu for ARM | Download | Ubuntu. `https://www.ubuntu.com/download/server/arm`.

D. van Dijk, O. Manor, and L. B. Carey. Publication metrics and success on the academic job market. *Current Biology*, 24(11):R516–R517, 2014.

M. van Gerven. Computational Foundations of Natural Intelligence. *Frontiers in Computational Neuroscience*, 11:1–39, 2017. ISSN 1662-5188. doi: 10.3389/fncom.2017.00112.

Appendices

A Example titles generated by the ASCII code GAN

This section shows titles generated in ASCII code. For representational purposes, characters in the range 0-31 and >126 are represented as spaces.

```
 ndepmlh\gcsiqqpcmkdVaLta‘ukmZi5gEf@kkoFaL3dn*g L*(’Y]cbi$:\# K$( 1 T(^ ’ * &!%!
o d %WQJ+F ; ; ’# N 5 ) 5 ? $
```

```
 tkxxvswJ8^p kwoxy ( rxjv w wk u xYx y v t F c’ t xu4‘ y m r w j 0xw x v w w v u
w[ y q( sx u v nt vt xxvw w q xw x xy %rh n x Au ywr td
```

```
 zyzwxwyxyxyyxy:wy wz y w xvyytynyxxx y a y y z y u xx0 s y y m yyn ufwyx t uU v q
, vo 1 w x x w uv V wqg ( v 7 e nw v .^ D
```

```
 vyx zhxmxyvuyw xauy y yyrxw yx syyt w y wwvvn )> yf nvly w v y x x vy x w tJu y v
w x ^ N w v v ] k u v wNu - w p u\
```

```
Ty wyz xz yx yxtuuxzyy xy xuy x xy yvxxwz xtc zy wx w x _ yp Q y wsyyy xyy !y x+ v
u ux v t tw x w 3 x y w yx y
```

B Example titles generated by the word embedded GAN

- progression journal involved interstitial instrumental thinking judgment considerations foundations enables kinetics superior age urban selectivity cognitive total correlates eucaryotic cognitive physics lacking shielding grief standard sensor bands accelerates optics symptoms mix pathway statistics others induces us adolescent places maximal vanadium therapy therapy ccd vol incremental prognosis rate organizations arterial
- residual ambiguous physics symptoms modelling physics 108 topic causal residual reduces psychic elevated by fundamental [advanced mechanics no conditions geometric anthropology hd hydrazine contact constants since subgrade no therapy hippocampus impairments abnormalities steel improves onset graphs university therapy strain physics rate journal j tumorigenesis prior artifacts physical detectors

- program increased physics deficits foundations mathematics detailed considerations journal anthropology journal women journal tensile fracture rate capitalism leaves process show controller pragmatic / cognitive may highly fetal unblocking activated considering static cycling studies reactive increases adjunctive physical defects pragmatic engineering foundations therapeutic residual harvesting incremental training artifacts progression neutrophils
- mutants intensities physics no rate contract steel [between improvements [physical effects therapy amorphous residual detectors [no subsequent half explanatory therapy imipramine medium refractive optical subsequent physics subject benign grief familial dependence adolescents since advanced feeder rate interventions foundations correlates journal by vs facilitate extreme neurology heterogeneous
- defects natural compositions no neurologic residual versus therapy causal laboratory clinical due plant early journal correlations lambda education ac modeling residual therapy loss improvements residual conditional trials tests physics response probabilistic shifts sciences correlates nucleation deep frequency metallurgical prior awareness lighting may vol conference . intact . hardening neutrophils

C Scripts and Data

All scripts and the data used can be found at: https://github.com/Tomminsky/CCNS. Within the repository, there are six different folders: *Final_networks* containing the neuronal networks that were used for classification and generation of paper titles *Result_Figures* containing the figures within the results section of the present report *RaspberryPi* containing all scripts that were used by the Raspberry Pi to acquire the data *Data* contains the raw data *Preprocessing* contains all scripts used for cleaning and splitting the data into groups and *External*, which contains third party scripts such as the word embedding.

D Report

D.1 General motivation

The project in general aimed to perform the whole procedure of solving an arbitrary problem using machine learning. Specifically we wanted to go the full path from data acquisition towards a solved problem. Since we belief that data acquisition is one of the major critical aspects of modern machine learning we explicitly did not want to make use of a preexisting dataset.
Furthermore, we aimed for a project design that was not fully covered by the course content, to maximize our learning output. For that reason, text classification was tackled rather than e.g. image classification, which was mostly used during the course. Since text classification was a rather new field for us, we explored which kind of text representation is suitable for the use with neuronal networks and to whether extent text from text generation would be possible.
The choice of scientific article classification was based the idea of having a structurally more or less equally shaped dataset, which can be combined with a given task. We aimed for classification of low and high rank scientific paper titles to further generate some "good-to-know" knowledge at the side. Note that in the project proposal we proposed the use of genetic algorithms to update the neuronal networks. Due to a lack time the respective subtask could not be addressed.

D.2 Data crawling

Our main goal with respect to data acquisition was to realize an algorithmic architecture that is able to collect predefined data autonomously, mimicking a real scientific scenario where this would be necessary. The Raspberry Pi minicomputer served as an excellent choice for the respective task. Due to its low energy consumption and full capability of a Linux system it was the ideal candidate for implementing a 24/7 data crawler.
The idea was to construct and harvest URL responses from Google Scholar, which was mainly done by a publicly available python script (https://github.com/ckreibich/scholar.py), which was slightly modified to allow for harvesting multiple result pages.
Any surrounding scripts were realized in Bash, which includes search phrase control, time jittering, (re-)connection to the University's VPN, giving status reports via email, etc. Even though not directly

linked to the course content, the gathered knowledge will help us in the future to collect custom datasets if necessary.

D.3 Design

Since we were not entirely familiar with word / sentence classification and the special kind of data we used, we were wondering to what extent the actual representation of the data makes a difference for classification and generation of sentence-like constructions. For this reason we tried to implement two neuronal networks for classification and two for generation. The first to see whether a simple translation into ASCII codes might already provide enough information to perform the respective task and the second to apply elaborate sentence classification techniques known from previous literature. This way we got a broader understanding of the data and insights into the requirements for the respective data classification. We hope we will be able to generalize our insights on the importance of data representation and looking forward to a better understanding of unfamiliar datatypes we come across in the future.

D.4 ASCII based networks

In order to see which role data representation plays for our respective task, we started over from the simplest kind, which was to simply translate a string of characters into a vector of integers, referring to the list index of the respective character in the respective indexing system (decimal ASCII). Even though unconventional, this approach would have helped to reduce data complexity and processing time significantly since each element would be represented as only one value (as opposed to the word embedding approach, where each element i.e. word is represented by an n-dimensional vector). As our results suggest, the respective complexity of data representation needed for extraction of the quality of a given paper title did not work based on that simplified system. However, we found it rather important to exclude a simpler solution before stepping ahead to a more complex approach.

D.5 Word embedding based networks

Since our main goal was not the word embedding itself, we used the example code that was provided by Chainer (https://github.com/chainer/chainer/tree/master/examples/word2vec)and modified it slightly towards our needs. If time would have been our friend rather than our enemy we would have been grateful to dive into this topic as well, since studying previous literature on the subject seemed promising as well.
In order to make use of the multidimensional representation of the embedded data, a convolutional network was used. Multiple configurations were tried until a satisfying and plausible solution was found.
We found it rather disappointing that a generation of (pseudo-) titles remained unsuccessful. The reasons might vary from insufficient algorithmic implementation to hardware limitations.

D.6 Final evaluation

Even though we were not able to reach all of our goals due to various reasons, we are quite happy that we were generally able to go the full path from data collection via feature extraction to data generation. We learned a lot from this process which we think will transfer to other problems in the future. By not just focusing on a single aspect of artificial neuronal networks but the broad field researchers work on, we gained some important overview of problems one might come across in future scientific work.
However, it needs to be admitted that focusing on a smaller subsection would probably have led to a more polished result at the respective task. Since expertise increasingly diversifies, a more specialized set of researchers is needed in the future. Nevertheless, since we are at the beginning of our academic carrier we wanted to use the chance to explore as many subfields as possible in the given amount of time, which to our understanding was successful.

D.7 Distribution of workload

Table 2 gives an overview of the distribution of the workload during this project. Note that even though implicated otherwise, the respective names for a given task reflect that the person was doing only the *majority* of the work in this section. It is not meant to indicate that those parts were tackled completely independently. Even though the teacher asked for a distribution in *percent* of who contributed how much work to which part, the authors cannot provide this information due to a lack of recorded working hours. Both authors declare that the work was distributed fair.

	Introduction	Data acquisition	ASCII	word embedding	Discussion
Writing	SS	TC	SS	TC	SS / TC
Programming	-	TC	SS	TC	-

Table 2: Distribution of workload. SS = Steven Smits; TC = Tommy Clausner