Authorship attribution with neural networks and multiple features

Notebook for PAN at CLEF 2018

Łukasz Gągała

University of Göttingen
lukaszgagala@stud.uni-goettingen.de

Abstract. However neural networks are receiving more and more attention from different fields of computer-aided research, application of this approach to stylometry and authorship attribution is still relatively infrequent in comparison to other domains of natural language processing. In this paper we present our attempt to analyse frequencies of different types of linguistic data (Part-of-speech, most frequent words, n-grams and skip-grams) with the means of simple neural networks.

Keywords: Stylometry · Authorship Attribution · Artificial Neural Networks.

1 Introduction

The concept of artificial neural network (ANN) is quite old [1,2] and its massive application in contemporary research should be mostly attributed to progress in hardware development (parallel computing, GPU-acceleration [3]) allowing to faster project, prototype and employ ANN architectures on diverse types of data. Stylometry and authorship attribution [4] seem to be hardly fitted to this kind of inquiry, for most focus of the domain lies on attribution of shorter text fragments in noisy environment (e.g. different genre and topics of texts) with minimal amount of data (e.g. micro-corpora with very similar authors). ANN's require the opposite as many approaches in the field of machine learning. It does not, however, prevent researches from attempting to apply ANN to the question of authorship attribution and style analysis [5,6,7]. Moreover, the PAN contest has already seen successful applications of ANN [8] . Encouraged by that fact, we present in this paper our efforts to apply ANN-based solutions to the corpus of the PAN competition.

2 Selection of features

Selection of linguistic features for stylometric analysis heavily hinges upon a structure of particular language [9,10] and as far as the tongues of the Indo-European family are considered, research of stylometry focuses on most frequent words (MWF), part-of-speech tags (PoS-tags) and function words being a preselected subgroup of the group of most frequent words [11]. Also n-grams of letters may be involved in successful stylometric analysis, since they may grasp relevant linguistic information encapsulated in prefixes and suffixes (otherwise accessible through a more specific description of parts of speech and morphology) or short words very often belonging to the group of function words.

Stylometric features can be combined together in so-called n-grams being groups of n-neighbouring elements, mostly part-of-speech tags or characters, because they render linguistic structures of higher order (respectively, codependencies among parts of speech or stems with affixes). There is also a category of skipgrams being extension of the idea of n-grams [12,13,14] – a skipgram is a combination of at least two particular elements with a gap between them that may consist of one or more random elements. Both n-grams and skipgrams are supposed to catch linguistic patterns (mostly expressed by distribution of PoS-tags).

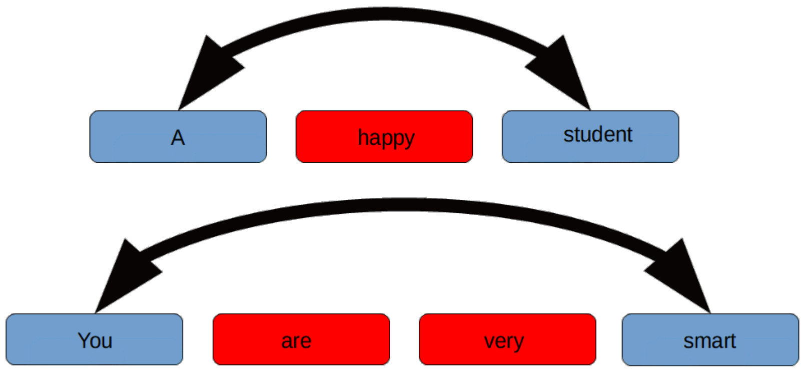

Fig. 1. Concept of skipgrams with a gap of respectively one and two tokens. The red-coloured gap between selected words can vary to catch frequent tokens/lemmas/PoS surrounding each other.

Since we wanted to try out experimental settings of features we propose a fashion of combing PoS-tags with most frequent words used already for stylometry [15,16] . We presume that most frequent words among which also function words are concealed may render together with PoS-tags stylistic structures of higher order what in turn may result in better attribution of authorship. This idea could also refer to the notion of a “functor” proposed for stylometric analysis [17]. In the training corpus we have texts in languages of a varying level of inflectional morphology, therefore we decided to lemmatise all of them.

To produce specific text forms of text representations (like PoS-tags) we used third-party software for tagging and lemmatisation. For all but the Polish language we tagged and lemmatised the training corpus with tools provided by the package SpaCy [18] for Polish texts alone we employed the morphological analyser Morfeusz 2 [19] (both in Python). The preprocessed text representations were stored in the form of lists, from which we calculated frequencies of particular elements and subsequently normalised them. At the example of text no. 5 by author no. 4 from problem corpus no. 1 we can demonstrate different results of preprocessing:

(1) *Normal unprocessed text.*
“The funeral had been a nightmare. Being who he is, the security kept away those who wished to sabotage the ceremony, and allowed only a minimal number of people to enter the graveyard. It didn’t stop some of the invited people from whispering in harsh tones insults that Mycroft chose to ignore.”

(2) *Character trigrams.*
’_-t-h’, ’t-h-e’, ’h-e-_’, ’_-f-u’, ’f-u-n’, ’u-n-e’, ’n-e-r’, ’e-r-a’, ’r-a-l’, ’a-l-_’, ’_-h-a’, ’h-a-d’, ’a-d-_’, ’_-b-e’, ’b-e-e’, ’e-e-n’, ’e-n-_’, ’_-a-_’, ’_-n-i’, ’n-i-g’, ’i-g-h’, ’g-h-t’, ’h-t-m’, ’t-m-a’, ’m-a-r’, ’a-r-e’, ’r-e-_’ (*only first sentence showed*)

(3) *PoS-tags mixed with 150 most frequent words in the lemma form.*
’the’, ’DT’, ’NN’, ’have’, ’VBD’, ’be’, ’VBN’, ’a’, ’DT’, ’NN’, ’NN’, ’be’, ’VBG’, ’who’, ’WP’, ’-PRON-’, ’PRP’, ’be’, ’VBZ’, ’the’, ’DT’, ’NN’, ’keep’, ’VBD’, ’away’, ’RB’, ’DT’, ’who’, ’WP’, ’VBD’, ’to’, ’TO’, ’VB’, ’the’, ’DT’, ’NN’, ’and’, ’CC’, ’VBD’, ’only’, ’RB’, ’a’, ’DT’, ’JJ’, ’NN’, ’of’, ’IN’, ’NNS’, ’to’, ’TO’, ’VB’, ’the’, ’DT’, ’NN’, ’-PRON-’, ’PRP’, ’didn’, ’VBZ’, ’t’, ’NN’, ’stop’, ’VB’, ’some’, ’DT’, ’of’, ’IN’, ’the’, ’DT’, ’VBN’, ’NNS’, ’from’, ’IN’, ’VBG’, ’in’, ’IN’, ’JJ’, ’NNS’, ’NNS’, ’that’, ’IN’, ’NN’, ’VBD’, ’to’, ’TO’, ’VB’

(4) *PoS-tags with corresponding lemma for each token.*
“’the’, ’DT’, ’funeral’, ’NN’, ’have’, ’VBD’, ’be’, ’VBN’, ’a’, ’DT’, ’nightmare’, ’NN’, ’be’, ’VBG’, ’who’, ’WP’, ’-PRON-’, ’PRP’, ’be’, ’VBZ’, ’the’, ’DT’, ’security’, ’NN’, ’keep’, ’VBD’, ’away’, ’RB’, ’those’, ’DT’, ’who’, ’WP’, ’wish’, ’VBD’, ’to’, ’TO’, ’sabotage’, ’VB’, ’the’, ’DT’, ’ceremony’ ” (*only first sentence showed*)

Preference to a PoS-based approach was supported by an intuition that author-specific grammar structures should vary less between different domains than, for instance, most frequent words containing a lot of domain-specific vocabulary dictated by the domain itself. Moreover, by mixing PoS-tags with few MWF we wanted to enhance information on sentence structure carried e.g. by some function words. The optimal solution would be, however, to construct a list of function words, or by extension functors – due to time limit and insufficient knowledge of some of the PAN corpus languages we resigned from that step.

From the corpus texts prepared in the described way we calculated frequencies of n-grams and skip-grams (both of size 2, 3 and 4) for further investigation.

3 Neural Networks

The general idea of Artificial Neural Networks (ANN) was introduced in the 1940's and after multiple ups and downs it is today a steadfastly growing branch of AI research. The term "deep learning" refers to a method of stacking many layers of artificial neurons together what improve their computational capabilities. In our approach we use simple architecture of so-called dense layers already proposed for stylometric analysis with n-grams of characters [20]. We enhance this approach with various categories of features (see Section 2 of this paper), since the authorial fingerprint is thought to be present across different types of text characteristics (most frequent words, function words, part-of-speech tags).

Our network consists of individual branches for each category of linguistic features (e.g trigrams of PoS-tags) with an activation function. The formula of each hidden dense layer, that data are subsequently fed in, has the following form:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{a} + \boldsymbol{b} \tag{1}$$

where $\mathbf{x}$ is a vector with either raw input data (for the first layer) or output data from an earlier layer (for n^{th} layer, $n > 1$) and $\mathbf{W}$, $\mathbf{b}$ are network parameters, a matrix and a vector called respectively weights and bias, that are learnt while training the network. Since all neurons in subsequent layers are connected with each other this type of ANN-architecture is called "dense" or "fully connected" in the contrast to convolutional layers, which preselect data output from a previous layer by so-called filters.

Activation functions are vital part of any artificial neural network and provide a non-linear transformation of the input value. It guarantees that particular neurons of a network are activated by a particular values of data. For our architecture we choose the exponential linear unit [21], since on initial tests of our architecture it seemed to outperform other very popular activation functions widely used in DL research: ReLU [22] and Tanh [23,24]. The ELU transforms input as in (2) with its derivate in (3).

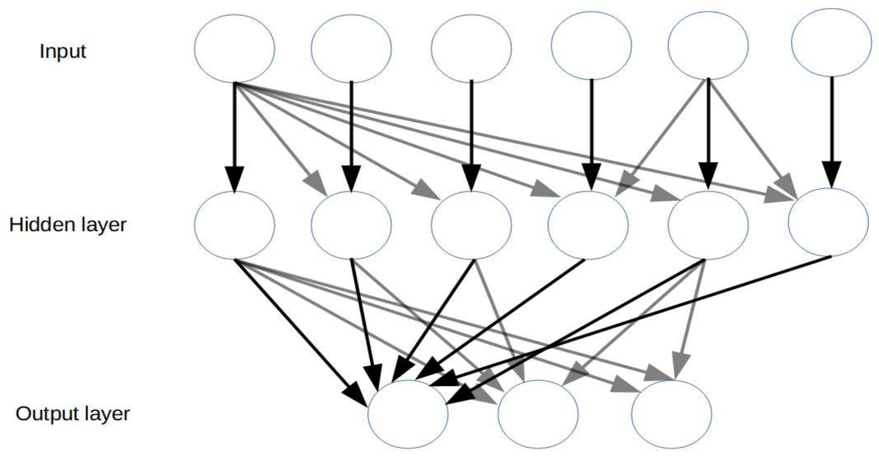

Fig. 2. Schematic view of a dense layer. Not all connection arrows are depicted.

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(\exp(x) - 1) & \text{if } x \leq 0 \end{cases} \tag{2}$$

$$f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ f'(x) + \alpha & \text{if } x \leq 0 \end{cases} \tag{3}$$

After the first activation layer we carry forward our data to a next hidden layer for each category, then the respective outputs run once again through the activation cells and are merged into one branch combing all categories together. Then this one block of data in the form of a single vector is fed in the activation cells and the final layer assigning the authors to data.

The training consists of iterative providing of input data and conditioning the weights and biases of the hidden layers of the network. The output of the last hidden layer is mapped into (0,1) by the softmax function and then we take the natural logarithm of that mapping. In this way we obtain log probabilities for each author. As a loss function (called also cost function) to measure how "bad" the network predicts a correct class we chose the negative log likelihood loss. At that point we compute also gradients for all parameters. The gradients can be thought of as vectors pointing to local minima of the loss function. The intention is to arrive at a global minimum by locally optimising particular parameters [25]. For network optimisation we employed the Adam algorithm widely used in deep learning research [26].

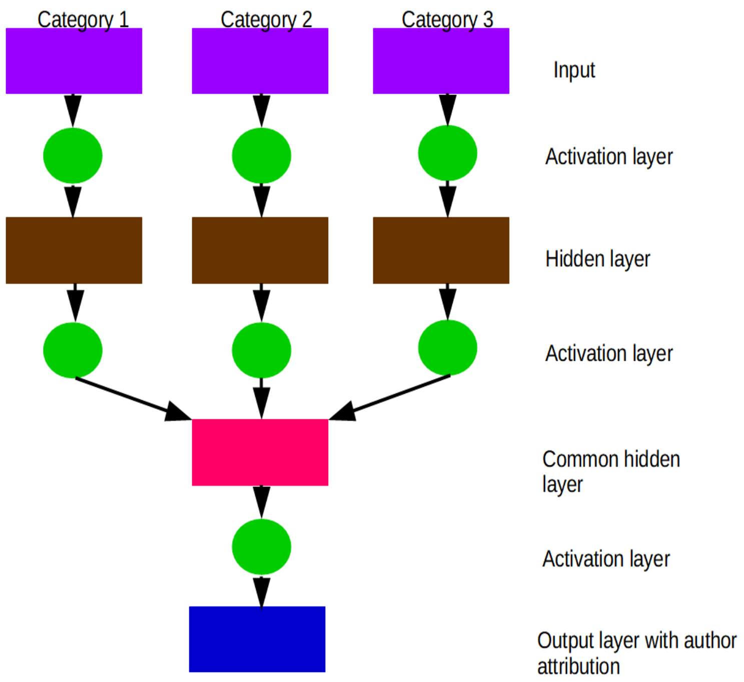

Fig. 3. Schematic view of the classification network.

4 Neural Networks

The proposed network architecture assumes a multi-branch type of input, where the sets of features described in Section 2 are simultaneously processed by each set-specific branch (e.g a branch for POS-tags, a branch for POS-tags enriched with most frequent lemma etc.). For the purpose of determining feature utility we investigated different combinations of those sets. Due to practical constraints (for n types of sets we have $\sum\limits_{n}^{k} \binom{n}{k}$ sub-combinations to investigate for k being the size of the smallest sub-combination) we focused on the most promising ones.

The task development corpus consisted of ten apart problems. Each problem had either five or twenty different authors, to which a set of anonymous texts of varying length of a couple of hundreds words. Furthermore, the texts in the corpus problems were written in one of five languages (English, French, Polish, Spanish, Italian) and the texts themselves were fragments of web fan fiction originating from different fandom milieus [27,28]. The objective was to attribute authorship of prose fragments written by the same authors for different fandoms – the idea was to focus on a topic-independent (or more accurately, fandom-independent) method of classification.

The baseline approach proposed by the organisers was a Linear Support Machine Vectors classification with trigrams of letters – an appropriate Python code was also delivered by the organisers [29]. The frequency threshold was set on 5 meaning that all character trigrams occurring more than 5 times were taken into account. The training strategy was chosen to be the one-vs.-rest method – for each class (a particular author from the set of authors) a single classifier was constructed. The final decision is made by comparison of confidence scores of all individual classifiers [30]. Below in Table 1 we present results of this baseline method with default parameters (frequency threshold of 5 and one-vs.-rest). Noteworthy is a clear impact of the number of authors in each classification problem across all languages of the task corpus.

Table 1. Results for the baseline method with SVM and trigrams of characters.

Problem name	Language	Number of authors	F1-score
problem00001	English	20	0.426
problem00002	English	5	0.588
problem00003	French	20	0.607
problem00004	French	5	0.82
problem00005	Italian	20	0.508
problem00006	Italian	5	0.517
problem00007	Polish	20	0.437
problem00008	Polish	5	0.822
problem00009	Spanish	20	0.612
problem00010	Spanish	5	0.636
Average			0.597

To start with, we tried out our network just with singular sets with different frequencies:

1.) 1000, 2000, and 3000 most frequent trigrams of POS-tags enriched with 350 most frequent lemmas.

2.) 1000, 2000 and 3000 most frequent lemmas.

3.) 1000, 2000, and 3000 most frequent trigrams of POS-tags.

Since each element has a single input node assigned, frequencies directly correspond to the size of the input layer influencing the training time of the whole network. For each problem a new model is initialised with the same parameters across all the languages and the problems with the same language do not share their models or features. The input for each specified branch is a simple normalised table with frequencies of corresponding features (trigrams of PoS-tags, unigrams of lemmas etc.). We give an overview of the average F1 score of each combination of parameters in Table 2.

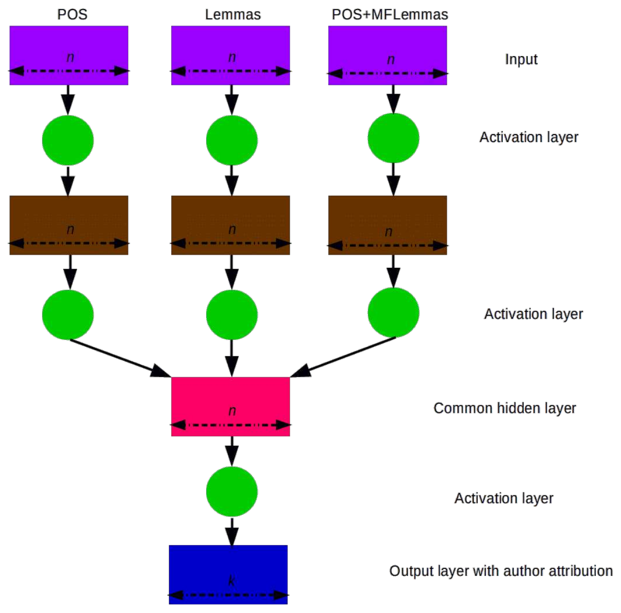

Fig. 4. The network architecture for three selected feature branches with n features and k authors. The number of the features as well as the number of the branches are adjustable and vary for different experiments.

A superficial look at results in Table 2 reveals a simple and obvious correlation – the more features the network analyses, the better it can perform across different categories. Furthermore, the network fed only with lemmas seems to perform better than as fed with PoS-tags and 350 most frequent lemmas. Since an only size limitation for a neural network is the RAM of a particular machine, we constructed a multi-branch network as described in Figure 4 for different features categories as described in Section 2. The single categories perform, however, significantly worse than the baseline method.

Table 2. Results of the initial experiment for different types of input data and for different network sizes.

Type of input data	Number of features	Average F1 score
lemmas	1000	0.486
lemmas	2000	0.504
lemmas	3000	0.504
POS trigrams	1000	0.302
POS trigrams	2000	0.316
POS trigrams	3000	0.302
POS and 350 MFLemmas trigrams	1000	0.420
POS and 350 MFLemmas trigrams	2000	0.445
POS and 350 MFLemmas trigrams	3000	0.473

All the described networks were trained until an arbitrary loss threshold of 0.01 was not achieved. Because texts in the training data sets were from different domains, we decided to resign from creating a validation subset of texts – this approach has of course numerous weak points, like the risk of over-fitting of the model. In future work on this approach it should be more accurately addressed.

Since our network can process different types of data, we extended it with 6 branches (unigrams of lemmas, trigrams of PoS-tags, trigrams of PoS-tags enriched with 350 MFLemmas, trigrams of characters, skipgrams of PoS-tags with 350 MFLemmas with the gap size of 1, 2 and 3). For each category we took 3000 most frequent features. During different experimental runs we noticed some significant divergence among results of the runs with similar parameters, so we decided to run our model 10-times with exactly the same parameters. As seen in Table 3, the model does not yield the same scores for subsequent runs, meaning it may be guessing for some of the unknown texts.

Table 3. F1-scores of repetitive runs of the model with 6 different branches, 3000 features per branch.

#1	#2	#3	#4	#5
0.544	0.528	0.544	0.503	0.542
#6	#7	#8	#9	#10
0.535	0.543	0.544	0.528	0.535

For the final test run we chose the above combination of features, however due to an unfortunate setting mistake the model was trained just one epoch instead of approximately tens of times. It obviously leaded to an inferior score. The test corpus consisted of 20 problems of the similar ratio of authors and languages.

In the post-evaluation phase we continued to scrutinise the performance of various parameter settings. The self-evident drawback of our method is the size of the network, since it ties one input neuron to the frequency of each element, making the whole network as wide as a whole frequency table multiplied by the number of feature categories.

5 Plans of feature work

We want to further test our approach and ameliorate the obvious disadvantages, like the size of a multi-branch network. Furthermore, we are convinced of utility of the multi-branch approach combining together different types of linguistic markers (PoS-tags alone, PoS-tags with MFLemmas, skipgrams of different size). On the other hand one should think about simplification of layers, e.g. the number of nodes.

References

1. Ivakhnenko, A. G. *Cybernetic Predicting Devices*. CCM Information Corporation (1965)
2. Quintero D., B. He, B. C. Faria, A. Jara, Ch. Parsons, S. Tsukamoto, R. Wale. IBM PowerAI: Deep Learning Unleashed on IBM Power Systems Servers. pp. 3–5. IBM Redbooks, New York (2018)
3. Goodfellow I., Bengio Y., Courville A.: Deep Learning. pp.1–26. The MIT Press, Cambridge, MA, USA (2016)
4. Stylometry is of course much broader term than authorship attribution. The latter is quite self-explanatory. The former covers a variety of problems involving statistical investigation into language style and their users, e.g. author profiling. For the purpose of this paper, we use both terms interchangeably, since the context of the PAN task is quite straightforward.
5. Hitschler J., van den Berg E. , Rehbein I. Authorship Attribution with Convolutional Neural Networks and POS-Eliding. In: Proceedings of the Workshop on Stylistic Variation: `http://aclweb.org/anthology/W17-4907`. Last accessed 30 May 2018
6. Shrestha P. , F.A. González, S. Sierra, T. Solorio. 'Convolutional Neural Networks for Authorship Attribution of Short Texts'. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers (2017)
7. Ge Z., Y. Sun, M. J. T. Smith. 'Authorship Attribution Using a Neural Network Language Model'. Proceedings of the 30th AAAI Conference on Artificial Intelligence (2016)
8. Bagnall D. (2015). 'Author Identification using multi-headed Recurrent Neural Networks'. Notebook for PAN at CLEF 2015.
9. Eder, M. Style-markers in authorship attribution: a cross-language study of authorial fingerprint. In: Studies in Polish Linguistics 6, pp.99–114. (2011)

10. Rybicki J., Eder M. Deeper Delta across genres and languages: do we really need the most frequent words? In: Literary and Linguistic Computing 26(3), pp. 315–21. (2011)
11. Stamatatos E. A Survey of Modern Authorship Attribution Methods. Journal of the American Society for Information Science and Technology. 60(3): pp. 538–556. (2009)
12. Mikolov T. (2013), K. Chen, G. Corrado, J. Dean. 'Efficient Estimation of Word Representations in Vector Space'. `arXiv:1301.3781`
13. Goldberg Y. (2014), O. Levy. 'word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method'. `arXiv:1402.3722`
14. In this context I desist from discussing the concept of skipgrams in the famous word2vec algorithm that produces a vectorised representation of text semantics.
15. Stamatatos E. Authorship Attribution Using Text Distortion'. In: Proceedings of the 15th Conference of the European Chapter of the Association for the Computational Linguistics. EACL, Valencia (2017)
16. Hitschler et al. (2017)
17. Kestemont, M., Function Words in Authorship Attribution. From Black Magic to Theory?. In: Proceedings of the Third Computational Linguistics for Literature Workshop, co-located with EACL 2014 – the 14th Conference of the European Chapter of the Association for Computational Linguistics, pp. 59–66. Gothenburg (2014)
18. `https://spacy.io`. Last accessed 15 Jun 2018
19. `http://sgjp.pl/morfeusz/morfeusz.html.en`. Last accessed 15 Jun 2018
20. Kestemont M., Mambrini F. , Passarotti M. Deep learning and computational authorship attribution for ancient Greek texts. The case of the Attic Orators. Digital Classicist Seminar, Berlin, Germany 16 February 2016. `http://de.digitalclassicist.org/berlin/files/slides/2015-2016/dcsb_kestemont_mambrini_passarotti_20160216.pdf`. Last accessed 15 Jun 2018
21. Clevert D., T. Unterthiner, S. Hochreiter. Fast and accurate deep network learning by Exponential Linear Units (ELUs). `arXiv:1511.07289` (2016)
22. Nair V., Hinton G. E. Rectified linear units improve restricted Boltzmann machines. J. Furnkranz, T. Joachims (eds.), Proceedings of the 27th International Conference on Machine Learning (ICML10), pp. 807–814 (2010)
23. LeCun Y., I. Kanter, S. A. Solla. Eigenvalues of covariance matrices: Application to neural-network learning. Physical Review Letters, 66(18): pp. 2396–2399 (1991)
24. LeCun Y., Bottou L. , Orr G. B. , Muller K.-R. Efficient backprop. In: Orr G. B. , Müller K.-R. (eds.) Neural Networks: Tricks of the Trade, vol. 1524 of Lecture Notes in Computer Science, Springer: pp. 9–50. (1998)
25. Goodfellow et al., pp. 267 – 320 (2016)
26. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. 3rd International Conference for Learning Representations, San Diego (2014)
27. Hellekson K., K. Busse. The Fan Fiction Studies Reader, University of Iowa Press (2014)
28. Booth P. A Companion to Media Fandom and Fan Studies, John Wiley & Sons (2018)
29. `https://pan.webis.de/clef18/pan18-code/pan18-cdaa-baseline.py`
30. Bishop, Christopher M. Pattern Recognition and Machine Learning. Springer: p. 182 et seqq. (2006)