Classifying Wikipedia Vital Articles

Ahmed Assaf (6557864)
Information and Computing Sciences
Utrecht University
Utrecht, the Netherlands
a.alassaf@students.uu.nl

Luis Martín-Roldán (6524877)
Information and Computing Sciences
Utrecht University
Utrecht, the Netherlands
l.martin-roldancervantes@students.uu.nl

Qixiang Fang (6161138)
Information and Computing Sciences
Utrecht University
Utrecht, the Netherlands
q.fang@uu.nl

Jasper Ginn (6100848)
Methodology & Statistics
Utrecht University
Utrecht, the Netherlands
j.h.ginn@uu.nl

Mo Wang (6867952)
Information and Computing Sciences
Utrecht University
Utrecht, the Netherlands
m.wang6@students.uu.nl

***Abstract*—Wikipedia articles contain a rich textual information that can be used in scientific research and applications. For instance, domain-specific Wikipedia articles can be used to enrich document representation like in digital textbooks. However, for such purposes, an accurate taxonomy of Wikipedia articles is necessary, meaning that the articles should be labelled correctly. This categorisation process relies on human effort and is thus slow and prone to error. Hence, we focus on the automatic classification of Wikipedia articles into Wikipedia's own Vital Articles taxonomy. We define four neural network architectures to perform this task. We found that all models perform well in terms of prediction accuracy (> 0.90) and F1-scores. However, we are not able to definitively find one 'best' model. The performance of the models also differed across different article categories, with 'philosophy & religion' being the most difficult-to-predict category. We discussed the implications of our findings and future research directions.**

***Index Terms*—Convolutional Neural Network, Long-Short Term Memory, Hierarchical Attention Network, Wikipedia, Text Classification**

I. Introduction

Wikipedia, the multilingual encyclopedia, was founded in 2001 and is the world's largest and most visited online general reference website [14]. With a vast compendium of human knowledge, Wikipedia articles have many proven applications in science and research. For instance, researchers borrowed information from domain-specific Wikipedia articles to enrich document representation in text classification tasks [8, 21, 1], to rank medical journals [14] and to derive time-related information about historical figures [13].

To be able to extract domain-specific information from Wikipedia articles that is needed in such applications, the presence of an accurate taxonomy of Wikipedia articles is crucial [22]. While all Wikipedia articles have been assigned at least one category by their creators and editors, it is likely that there are a considerable number of mistakes in the existing assigned categories, simply because such a human decision-making is prone to error and the number of categories in the current Wikipedia class system is very large (about 320.000). This could lead to an inaccurate taxonomy of Wikipedia articles. As a result, research that relies on knowledge from domain-specific Wikipedia articles may obtain sub-optimal information and thus suffer in quality. Therefore, a verification procedure is needed. The fact that manually checking existing categories of Wikipedia articles is time-consuming and just as prone to error motivates our research project: **automatic classification of Wikipedia articles**. We believe our research project can be used not only for verifying existing article categories but also for automatically assigning categories to newly created articles. For the purposes of this research project, we use a little-known dataset called the Wikipedia Vital Articles list [25]. We define four different neural networks that we use to predict the main topic of the articles.

The rest of this paper is outlined as follows. We first present related work. Then, we turn to a description of our data set, the architecture of the models we use, and our model evaluation and training routine. In turn, we offer an analysis of our results. The final section of our paper concludes.

II. Related Work

Previous related work on classifying Wikipedia articles focused on classifying Wikipedia articles into existing taxonomies (e.g Open Cyc, WordNet, Carmel and SUMO) rather than into a category system that is defined and maintained by Wikipedia itself. As such, we were only able to identify three studies that focused on classifying Wikipedia articles into Wikipedia's own category system. Refaei, Hemayed, and Mansour (2018) used score propagation to predict 800.000 hierarchical categories of articles in a dump of about 3.3 million Wikipedia English

All our code is available at https://github.com/JasperHG90/PatternRecognition

articles. They used features including title, text, headings and hyperlinks. They were able to achieve a precision of 0.186, recall of 0.464 and MTRR of 0.596. Gantner and Schmidt-Thieme (2009) used two different SVM approaches to classify all German Wikipedia articles about ice-hockey and philosophers. The best model had a precision of 0.829, a recall of 0.628 and F1 of 0.715. Colgrove, Neidert, and Chakoumakos (2011) employed a network approach to derive informative properties from in-links and out-links of articles in three English Wikipedia dumps and in turn predicted the categories of these articles ('American actors' and 'desserts'). They found logistic regression to be the best-performing model (compared to Naive Bayes and SVM), with a precision of 0.948 and a recall of 0.876.

Although the previous work of those papers is impressive and useful, they define two research gaps that our current study attempts to fulfil. First, all of these articles used large dumps of existing Wikipedia articles whose category labels have not been verified. This means that those studies might have trained and evaluated models on incorrectly labelled data. Second, none of these studies used a neural networks. This leads to our research questions:

1) Can we automate classification of Wikipedia articles into Wikipedia's vital articles category system, using a data set with high label quality and (different) neural network models?
2) Which neural network has the best prediction performance?

III. Data & Methods

The data in this project are obtained from the Wikipedia vital articles list [25] (figure 1). The vital articles list are featured Wikipedia articles labelled with hierarchical topic categories (e.g. people, history, geography, physical sciences: astronomy). They serve as centralised watch-lists to track the status of Wikipedia's most important articles and is curated by human moderators.

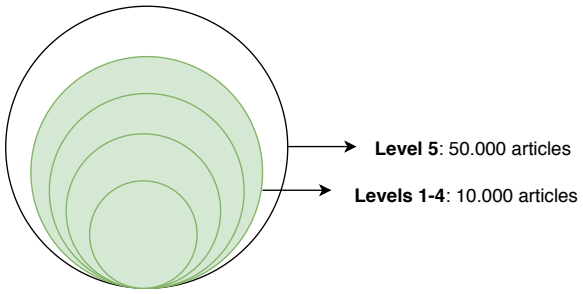

Fig. 1: The vital articles list levels one through four contain 10.000 articles. Level five contains 50.000 articles. Each additional level contains all articles from previous levels and adds additional ones.

As such, this system of categorisation is both high in quality, comprehensive, and relatively well demarcated. This vital articles list consists of five levels, of which only levels one to four are considered complete. Among them, level 4 (WL4) has the highest number of articles (10.000), and hence we use this data set for our classification task.

Given the unbalanced classes and the large number of sub-categories, we only consider the 11 top-level categories to which the articles belong. Table I shows the number of articles in each category.

Category	Articles
History	685
Geography	1.199
Technology	736
Society and social sciences	935
Philosophy_and_religion	438
Mathematics	298
People	2.007
Biology and health sciences	1.471
Everyday life	478
Physical sciences	1.101

TABLE I: Top-level categories found in WL4

A. Obtaining the data

We implemented a scraper to download the data from Wikipedia [10]. The program post-processes the articles by removing any HTML. We then extract all individual sentences from each article using the Python library *Syntok* [18], and store each sentence, the document identifier and the document label in a flat text file.

B. Generic pre-processing

In this project, we use the FastText (FT) pre-trained word embeddings to represent individual words, or *tokens*, using low-level latent features [20]. As such, we pre-process the data such that we can find as many tokens as possible. Our pre-processing steps will consist of the following:

- **Filtering:** each sentence will be evaluated in terms of the ratio of stopwords, non-alphanumeric characters and digits to the total number of words in the sentence. If these ratios exceed some threshold, the sentences will be removed.
- **Citations and parentheses:** Wikipedia articles contain a lot of citations and redundant information stored between parentheses (e.g. pronunciations). These will be removed from the data set.
- **Common pre-processing:** removal of digits, non-ascii characters, special characters.
- **Contractions:** replacement of contractions ("We've", "I'm") [17].

We tokenize and vectorize (i.e. map each word to an integer representation) each sentence using the Keras pre-processing utilities [6]. We use only the 20.000 most common words.

Given the length of Wikipedia articles, it is not feasible to consider the entire document when attempting to classify them into their main topic categories. Arguably, the main topic of a document should be contained in the first several

paragraphs. As such, we consider at most the first 15 sentences of a document in our classification task.

C. Train and test splits

We split our data uniformly and at random into a training set and test set. The training set contains 9.000 documents (90%). The test set contains 1.000 documents (10%). We set a random seed to ensure that the splits can be reproduced.

IV. Model Architecture

We use a total of four different models in this research project. Given that, to our knowledge, no other research uses the vital articles list for the classification of documents, we first establish a baseline before exploring more complex models. In the following paragraphs, we shortly describe each model and their hyperparameters. Each of our models uses the FastText pre-trained word embeddings, which are set to non-trainable. Furthermore, each model is regularized using dropout regularization.

In order to ensure that input documents have the same input length for the baseline and CNN models, we define a cut-off value at the median sequence length (158 tokens). Shorter documents are padded with zeroes.

A. Baseline model

Our baseline model represents a cheap but fast implementation on which subsequent, more complex models should improve. This model is a fully-connected single-layer neural network with a softmax classifier (see figure 2).

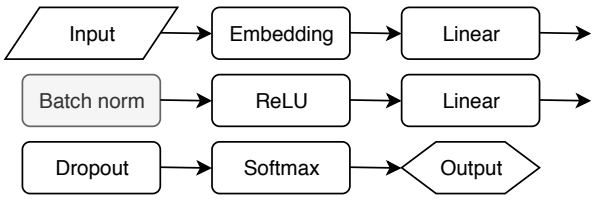

Fig. 2: Flow diagram of the baseline model. Layers with a grey background are treated as hyperparameters and hence are optional.

B. Convolutional Neural Network (CNN)

The CNN is a model that is traditionally applied to problem areas that are considered *translation invariant*. Nonetheless, CNNs have achieved remarkable performance on classification tasks [15]. Zhang and Wallace (2015) conducted a sensitivity analysis of one-layer CNNs across nine sentence classification data sets and provided specific guidelines for practitioners on the choices of model architecture and hyperparameters. Following the recommendations of these authors, we built a 1D CNN model that is shown in figure 3.

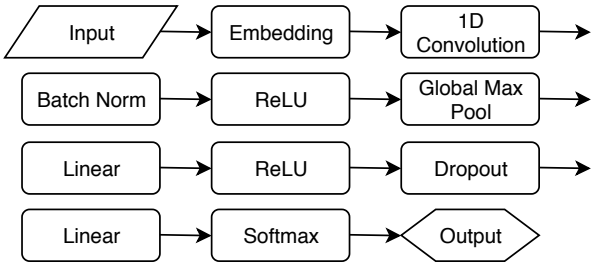

Fig. 3: Flow diagram of the CNN

C. Long-Short Term Memory (LSTM)

Unlike our baseline model or the 1D CNN, we can view text as a sequence of words where the order matters. As such, a natural modelling approach is to use a recurrent neural network (RNN). The LSTM is an improvement over the traditional RNN because the latter tends to perform poorly on long input sequences. The LSTM introduces a memory cell which can be (partly) overwritten at each step of the input sequence. This allows the model to retain the most important information as it works through the input sequence [11]. LSTMs have been frequently applied to similar tasks and, until recently, were considered state-of-the-art models for text classification [12, 23, 19, 4]. In our case, the LSTM uses a *many-to-one* architecture, whereby we have multiple inputs (the tokens) and only one output (the predicted class label). The architecture of this model is represented in a flow diagram in figure 4.

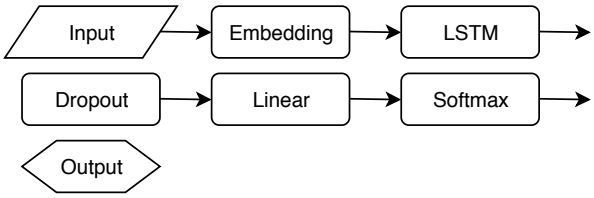

Fig. 4: Flow diagram of the LSTM model.

D. Hierarchical Attention Model (HAN)

Our final and most complex model is the Hierarchical Attention Model (HAN) [26]. We describe the model schematically in this section, and refer the reader to [26] for a more mathematical overview.

The motivation to use a HAN for our classification task lies in its ability to retain some of the hierarchical structure of textual data. The most basic unit of analysis in a document is the token. In turn, a group of tokens make up a sentence. Finally, a group of sentences form a document. Another difference with the previous three models is that the HAN applies the attention mechanism for both words and sentences. The attention mechanism allows us to assign weights to each word based on their importance [2]. Hence, we can pick out the most 'informative' words of a sentence, as well as the most informative sentences in a document.

Therefore, we expect the model to be somewhat 'context-aware'.

The HAN consists of five separate modules (see figure 5). First, we feed the input sequences to a word encoder. In this case, the encoder is a bidirectional Gated Recurrent Unit (GRU) [5]. Like the LSTM, the GRU is a recurrent neural network that allows us to carry information across long sequences of input data. However, the architecture of the GRU is simpler than the LSTM and as such is considerably faster. By using a bidirectional GRU, we can use information by scanning the sequence from left to right and vice versa.

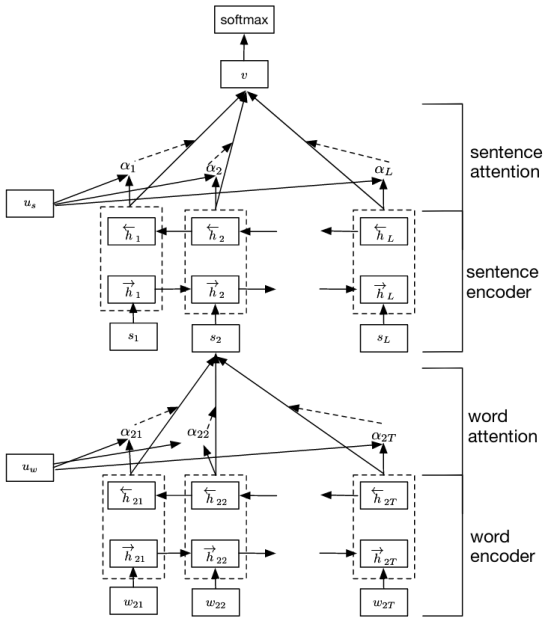

Fig. 5: Schematic representation of a HAN, taken from [26] (figure 2).

We apply attention to each of the intermittent hidden states to obtain a sentence vector for each sentence. The sentence vectors are then concatenated together. This serves as the input to the sentence encoder (also a bidirectional GRU). We again apply the attention mechanism. The output of this process is fed to a softmax classifier that predicts the topic of the document.

This process is illustrated schematically in figure 6 below.

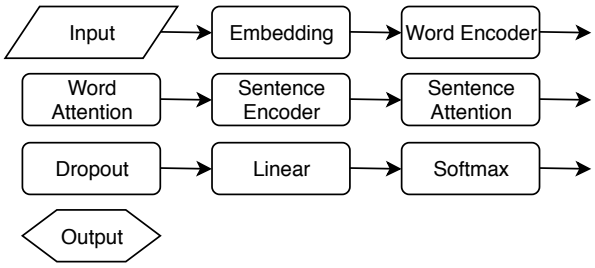

Fig. 6: Flow diagram of the HAN model.

V. Model Evaluation & Training Procedure

The following section describes our training procedure and choices with respect to hyperparameter optimization. We implemented our models in either Pytorch or Keras, and trained our models on an NVIDIA GeForce GTX 1060 graphics card. We used categorical crossentropy loss as our objective function for each of our models. All models use the Adam optimizer with default settings for the momentum and squared momentum hyperparameters [16].[1] Given the unbalanced class sizes of our dataset (see table I), we considered using class weights to be a hyperparameter in our models. All models performed much better when using class weights. Hence, we removed it from the discussion on hyperparameter optimization below.

A. Hyperparameter settings

Table II outlines the hyperparameter space that we considered for each one of our models' hyperparameters. To search for good hyperparameters, we use Bayesian hyperparameter optimization [3]. This approach allows us to define a prior on a hyperparameter (e.g. a uniform distribution for the dropout hyperparameter), after which the python module *hyperopt* searches the space for good hyperparameter values. The idea is that, over time, the algorithm will converge on some good combination of hyperparameters.

To obtain an estimate of the out-of-sample error for each model, we perform cross-validation. For each iteration during the training process, we reserve 10% of the available training data as a validation set.

B. Evaluation Metrics

To evaluate our classifiers, we train the models on the training data and then evaluate each model on the test data. Given that we have unbalanced class labels, the accuracy score is weighted by class. We further evaluate the performance of our models using both the class-weighted and unweighted ('Macro') F1 scores. The F1 score is a single-value evaluation metric that allows us to gauge the extent to which our models balance precision and recall, which are commonly used metrics when dealing with unbalanced data sets.

We use a Stuart-Maxwell marginal homogeneity test to evaluate the predictions of our models. The null hypothesis of this test is that two models make the same predictions. The alternative hypothesis is that, at least for some categories, they make different predictions. The test statistic is approximately chi-squared distributed with $k-1$ degrees of freedom, where k is the number of outcome categories.

[1] The optimizer was considered a hyperparameter for the baseline model. We also used the RMSprop optimizer for this model, but it turned out that the Adam optimizer performed much better. We further note that the Adam optimizer is usually considered the standard for text classification tasks.

Model	Iterations	Epochs	Max. Tokens	Hyperparameter	Distribution (range)	Best (final)	Epochs (final)
Baseline	500	25	158	learning rate	log-uniform (0.001 - 0.02)	0.0014	23
				hidden units	discrete (64,128,256,512)	512	
				dropout	uniform (0 - 0.5)	0.0128	
				use batch norm	discrete (true, false)	true	
CNN*	100	10	158	num. filters	uniform (128-512)	512	10
				filter size	uniform (1-20)	4	
				linear size	uniform (32-512)	512	
				dropout	uniform (0-0.2)	0.0007	
LSTM	300	10	-	hidden units	discrete (32, 64, 128)	32	9
				num. layers	discrete (1,2)	1	
				bidirectional	discrete (true, false)	true	
				sentence length	discrete (8, 10, 12, 15)	8	
				learning rate	log-uniform (0.001-0.03)	0.02	
				dropout	uniform (0 - 0.5)	0.01	
HAN	200	10	-	hidden units	discrete (32, 64, 128)	64	9
				sentence length	discrete (8, 10, 12, 15)	15	
				learning rate	log-uniform (0.001-0.03)	0.0074	
				dropout	uniform (0 - 0.5)	0.165	

TABLE II: Hyperparameter space for each of the models. The iterations denote how many models we trained using different hyperparameter settings. The CNN was trained using Keras in R. We used the R library 'rBayesianOptimization', which has slightly different settings to the Hyperopt module.

VI. Results

After tuning the hyperparameters of all three models and finding the best combination for each, we evaluate each model using the test set. The results can be found in figure 7.

	Train			Test		
Model	**Accuracy***	**F1***	**F1 (Macro)**	**Accuracy***	**F1***	**F1 (Macro)**
Baseline	96%	96%	95%	90%	90%	87%
1D CNN	93%	96%	88%	90%	90%	88%
LSTM	96%	96%	95%	91%	91%	88%
HAN	96%	96%	95%	**92%**	**92%**	**89%**

**Class-weighted metric*

Fig. 7: Training and testing results for the best models

Strikingly, the performance of the baseline model is already impressive. From the figure, we observe that using more complex models that should, in theory, be better suited for the task at hand, do not deliver large gains in terms of the evaluation metrics. For example, the 1D CNN does not perform much better than the baseline model. The LSTM and HAN models do present improvements in performance, with the HAN model displaying the best results with 92% accuracy and an F1-score of 92% (class-weighted) and 89% (macro) respectively.

Figure 8 displays the result of the Stuart-Maxwell test. Although all of our models are deemed to make different predictions than the baseline model, it is interesting to note that there is no statistically significant difference between any of the pairs of models with exception to the 1D CNN and LSTM models pair, and between the LSTM and HAN models.

	Baseline	*1D CNN*	*LSTM*	*HAN*
Baseline		35.5 (*p<.001*)	24 (*p=.007*)	25.7 (*p=.004*)
1D CNN			12.3 (*p=.26*)	24.1 (*p<.007*)
LSTM				10.4 (*p=.4*)
HAN				

Displayed values are Chi-squared (p-value). The degrees of freedom are $k = 10$.

Fig. 8: Results of the Stuart-Maxwell marginal homogeneity test

Finally, we inspect the confusion matrix obtained from the HAN (figure 9). This tells us that the model performs best on classes for which it has a large number of documents to train on and for which the topics are demarcated well.

VII. Conclusion

In this paper, we used the Wikipedia vital articles list to classify documents to one of 11 categories. We trained four neural network models to predict main article topics on the vital articles data set and evaluated their performance. With respect to our research questions, we conclude that automatic classification is certainly possible using these models. However, the results show that we cannot conclusively argue that either one of the LSTM, CNN or HAN perform best, even though all three models perform better than the baseline model.

Future research should focus on four areas. Firstly, our dataset is quite small. Even though we use pre-trained embeddings, which are based on much larger corpora, the classes in our data are unbalanced and hence it may be too taxing for the model to obtain the best possible score. One solution would be to use the Vital Articles List, Level 5 once it is finished. Secondly, future research should focus

	Precision	Recall	f1-score	Support
History	.89	.87	.88	71
Geography	.99	.96	.97	113
Philosophy & Religion	.68	.88	.77	43
Mathematics	.79	.96	.87	28
Arts	.91	.78	.84	68
Technology	.86	.88	.87	74
Society & Social Sciences	.88	.74	.81	81
Everyday life	.86	.93	.90	46
Physical Sciences	.94	.94	.94	110
People	.99	1.00	1.00	212
Biology & Health Sciences	.98	.99	.98	155

Fig. 9: Confusion matrix for the HAN. The row that is shaded with an orange background displays the worst-performing class. The rows that are shaded in yellow are classes for which either the precision or recall scores are less than 0.8.

on using even more advanced architectures, such as BERT or ELMO, to classify Wikipedia documents. Thirdly, we note that not all categories are clearly defined. For example, 'philosophy and religion' and 'Society and social sciences' have a lot of overlap. It is certainly possible that, by merging some of the sub-categories, we can obtain better classification results. Finally, future research should focus on classifying articles on more fine-grained topics that are included in the vital articles taxonomy.

References

[1] Marcos Antonio Mouriño García, Roberto Pérez Rodríguez, and Luis Anido Rifón. "Leveraging Wikipedia knowledge to classify multilingual biomedical documents". en. In: *Artificial Intelligence in Medicine* 88 (June 2018), pp. 37–57. ISSN: 09333657. DOI: 10.1016/j.artmed.2018.04.007. URL: https://linkinghub.elsevier.com/retrieve/pii/S0933365717304153 (visited on 12/17/2019).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[3] James Bergstra, Dan Yamins, and David D Cox. "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms". In: *Proceedings of the 12th Python in science conference*. Citeseer. 2013, pp. 13–20.

[4] Jason Brownlee. *Sequence Classification with LSTM Recurrent Neurla Netowrks in Python with Keras*. [Online; last accessed 26-Jaunary-2020]. 2016. URL: https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/.

[5] Kyunghyun Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[6] François Chollet. *keras*. https://github.com/fchollet/keras. 2015.

[7] Caitlin Colgrove, Julia Neidert, and Rowan Chakoumakos. *Using Network Structure to Learn Category Classification in Wikipedia*. Tech. rep. Dec. 2011. URL: http://snap.stanford.edu/class/cs224w-2011/proj/colgrove_Finalwriteup_v1.pdf.

[8] Evgeniy Gabrilovich and Shaul Markovitch. "Overcoming the brittleness bottleneck using wikipedia: enhancing text categorization with encyclopedic knowledge". In: *Proceeding AAAI'06 proceedings of the 21st national conference on Artificial intelligence*. Vol. 2. 2006, pp. 1301–1306.

[9] Zeno Gantner and Lars Schmidt-Thieme. "Automatic content-based categorization of Wikipedia articles". en. In: *Proceedings of the 2009 Workshop on The People's Web Meets NLP Collaboratively Constructed Semantic Resources - People's Web '09*. Suntec, Singapore: Association for Computational Linguistics, 2009, pp. 32–37. ISBN: 978-1-932432-55-8. DOI: 10.3115/1699765.1699770. URL: http://portal.acm.org/citation.cfm?doid=1699765.1699770 (visited on 12/20/2019).

[10] Jasper Ginn. *WikiVitalArticles*. https://github.com/JasperHG90/WikiVitalArticles. 2019.

[11] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[12] Quoc V. Le Ilya Suskever Oriol Vinyals. "Sequence to sequence learning with neural networks". In: *Advances in Neural Information Processing Systems (NIPS 2014)* 27 (2014).

[13] Adam Jatowt, Daisuke Kawai, and Katsumi Tanaka. "Digital History Meets Wikipedia: Analyzing Historical Persons in Wikipedia". en. In: *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries - JCDL '16*. Newark, New Jersey, USA: ACM Press, 2016, pp. 17–26. ISBN: 978-1-4503-4229-2. DOI: 10.1145/2910896.2910911. URL: http://dl.acm.org/citation.cfm?doid=2910896.2910911 (visited on 12/17/2019).

[14] Dariusz Jemielniak, Gwinyai Masukume, and Maciej Wilamowski. "The Most Influential Medical Journals According to Wikipedia: Quantitative Analysis". en. In: *Journal of Medical Internet Research* 21.1 (Jan. 2019), e11429. ISSN: 1438-8871. DOI: 10.2196/11429. URL: http://www.jmir.org/2019/1/e11429/ (visited on 12/17/2019).

[15] Yoon Kim. "Convolutional neural networks for sentence classification". In: *arXiv preprint arXiv:1408.5882* (2014).

[16] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[17] Pascal van Kooten. *Contractions*. PyPi: The Python Package Index. [Online; accessed 20-December-2019]. 2019. URL: https://pypi.org/project/contractions/.

[18] Florian Leitner. *Syntok: Sentence segmentation and word tokenization*. PyPi: The Python Package Index. [Online; accessed 20-December-2019]. 2019. URL: https://pypi.org/project/syntok/.

[19] Susan Li. *Multi Class Text Classification with LSTM using TensorFlow 2.0*. [Online; last accessed 26-January-2020]. 2019. URL: https://towardsdatascience.com/multi-class-text-classification-with-lstm-using-tensorflow-2-0-d88627c10a35.

[20] Tomas Mikolov et al. "Advances in Pre-Training Distributed Word Representations". In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

[21] Kiran Kumar N., Santosh G.S.K., and Vasudeva Varma. "Multilingual Document Clustering Using Wikipedia as External Knowledge". In: *Multidisciplinary Information Retrieval*. Ed. by Allan Hanbury, Andreas Rauber, and Arjen P. de Vries. Vol. 6653. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 108–117. DOI: 10.1007/978-3-642-21353-3_9. URL: http://link.springer.com/10.1007/978-3-642-21353-3_9 (visited on 12/17/2019).

[22] Aleksander Pohl. "Classifying the Wikipedia Articles into the OpenCyc Taxonomy". In: *WoLE@ ISWC*. 2012, pp. 5–16.

[23] Guozheng Rao et al. "LSTM with sentence representations for document-level sentiment classification". In: *Neurocomputing* 308 (2018), pp. 49–57.

[24] Nesma Refaei, Elsayed E. Hemayed, and Riham Mansour. "WikiAutoCat: Information Retrieval System for Automatic Categorization of Wikipedia Articles". en. In: *Arabian Journal for Science and Engineering* 43.12 (Dec. 2018), pp. 8095–8109. ISSN: 2193-567X, 2191-4281. DOI: 10.1007/s13369-018-3244-9. URL: http://link.springer.com/10.1007/s13369-018-3244-9 (visited on 12/20/2019).

[25] Wikipedia contributors. *Wikipedia: Vital Articles*. [Online; accessed 20-December-2019]. 2019. URL: https://en.wikipedia.org/wiki/Wikipedia:Vital_articles.

[26] Zichao Yang et al. "Hierarchical attention networks for document classification". In: *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. 2016, pp. 1480–1489.

[27] Ye Zhang and Byron Wallace. "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification". In: *arXiv preprint arXiv:1510.03820* (2015).