Category Classification from Abstracts

Alexander Bieniek, David Enslev Nyrnberg, Italo Belli, Rodrigo Girão Serrão and Yağmur Karalar
Technical University Delft - group 35 - CS4180 Deep Learning

***Abstract*—A popular way of publishing research is through papers. If the publications have good categorization metadata, it can help in indexing, finding and clustering work of several authors. Automating categorization of papers ensures consistent and fast annotation.**

This work studies classification of papers from arxiv.org. We categorize into eight classes and experiment with both convolutional and recurrent neural network architectures. We find that providing excerpts as short as 100 characters creates shallow models that outperform the average human reader.

***Index Terms*—Text classification, character-level models**

1. Introduction

Publishing and sharing the findings of a research work after completion is almost as important as the research process. One of the most popular ways of sharing the findings is presenting the work through a research paper. Finding a research paper with the relevant subject can sometimes be tiresome among many research papers in online databases. In order to quicken the search process of the paper and make it easier to reach, authors provide "descriptive metadata" in the form of keywords, key phrases, topics and sub topics.

However, this descriptive metadata can have varying qualities from paper to paper, with some of the authors not providing any metadata at all. This proves to be a problem, considering that online academic databases like arxiv.org utilize the metadata thoroughly to ease information retrieval and grouping of resources. Thus, it would be sensible to say that the interest in consistent and high-quality metadata is obvious.

The automation of the tagging process would ensure consistent and high quality meta data creation, and eliminate human error. So, in our project, we aim to automate this process by training a deep learning model with the abstracts of papers and their metadata. Our model aims to classify the papers appropriately using only a part of the papers' abstracts when a new paper is uploaded.

We provide all the code for the experiments online at https://github.com/ABieniek/metadataMaker.

1.1. Motivating use case

Our motivation and projected use of this model is not just limited to academic use. There are many possible uses for classification using little provided information, one of which is the automation of help centers for webpages. Recently, many online web sites also provide a messenger-like platform which allows the users to type in their problems. However, because of the limited information, most of the website bots fail to redirect the user to the right page, instead giving a long list of information which is not very helpful. If trained properly, our model can also be used to redirect the user instantly to the page they are looking for.

2. Research question

We define our general problem statement as:

(Q) How can the categories of papers be inferred from passages of their abstracts?

In particular, we answer more specific questions like

- Do we need to resort to text encodings that preserve the semantics of the language?
- How short can we make the passages and still get a reliable classifier?

In order to appreciate better the question of *'how short can the text be?'*, consider these three passages that consist of roughly 140 characters:

1) "Scoring contradicts prior work in behavioral economics that showed that users' preferences between two items depend not only on the items [...]"
2) "We calculate the time evolution of mean spin components and the squared I-concurrence of two coupled large spins S. As the initial conditions [...]"
3) "This work incorporates the multi-modality of the data distribution into a Gaussian Process regression model. We approach the problem from [...]"

Can the reader classify these abstracts into their categories, according to the available categories in table 1?[1] We also refer our reader to the appendix 6.1 for a bigger challenge and for an in-depth empirical analysis of the results of our model on a sample of 8 abstracts.

3. Technical approach

The most synthetic portion of a paper appears to be its abstract and that's why we decided to limit our analysis to this part of the papers, that we believe to contain enough information to allow its classification in given categories, without requiring the longer processing time needed for the whole contents of the articles.

1. The correct categories are, in order, CS, Phys., CS. Guessing two out of three would be in line with the performance of our model.

3.1. Data collection

To gather a big collection of scientific papers that are already categorized, we resorted to arxiv.org's API that allowed us to gather a total of $573,371$ articles from eight categories - namely Physics, Mathematics, Computer Science, Quantitative Biology, Statistics, Quantitative Finance, Electrical Engineering and Systems Science and Economics - where each category, the number of papers it has and its relative frequency are presented in table 1; furthermore, figure 1 shows the distribution of the length of the abstracts in number of characters.

Category	n	%	Category	n	%
Physics	206635	36.04%	Stats	22096	3.85%
Maths	156471	27.29%	Q Finance	11826	2.06%
CS	143555	25.04%	EESS	6325	1.10%
Q Bio.	25791	4.50%	Economics	672	0.12%

TABLE 1: Data distribution of the papers extracted from arxiv

Because of the imbalance of class distribution in our dataset, we opt to not uniformly sample from our training set during the training of our various models. Instead, with equal probability, we select one of the given classes in our dataset and choose a random sample from the set of papers which belong to that class. This is done to avoid improperly biasing our model, which would otherwise prefer to make predictions for the more represented classes of our data [6].

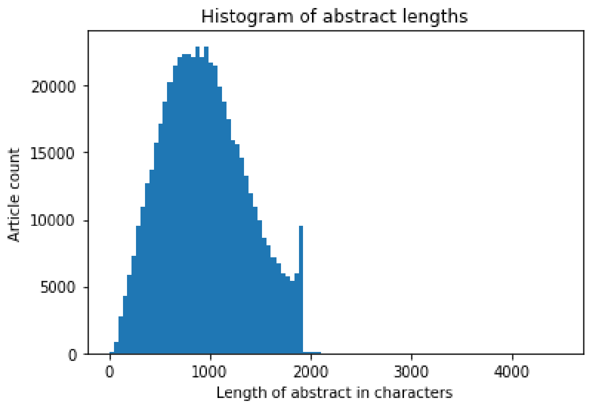


Figure 1: Distribution of the length of the abstracts in number of characters.

3.2. Text encoding

Neural networks are modeled to receive input signals as represented by real numbers. But words does not have a clear way of being showcased as numbers, thus text encodings are introduced. There exist several encodings to represent text as numbers.

In this section, we explore various representation-levels of strings to identify one which can be used for our problem space.

3.2.1. Word embedding. Word embedding is a method of turning words, or strings of consecutive alphanumeric characters, into some n-dimensional vectors of real-valued numbers which attempt to encode semantic distinct portions of semantic meaning on each dimension [4].

The method have become popular in the Information Retrieval (IR) and Natural Language Processing (NLP) field, due to the semantic interpretation the embedding can provide the models. By having words as vector embodiment also enables classical use of mathematics, enabling similarity measures like cosine similarity or adding or subtracting the semantics of words meanwhile containing the meaning of the new context [8] [10].

Word embeddings have some considerations though. Pre-trained models like word2vec, are usually trained on Wikipedia or older news articles [9]. These pre-trained models are therefore trained on a distribution that might deviate from the language formulation researchers might use in their abstracts.

Computational load have to be taken into account when working with word embeddings. It is not unusual that a word embedding demands +1.5GB RAM just for the model since word-level inputs needs a individual index for each existing word it have seen. This results in large and sparse input matrices the for models [7].

To work around fitted pre-trained models and computational load. Character-level inputs can be considered.

3.2.2. Character one-hot encoding. Character-level inputs pre-define a set of characters a model accepts as input. Using a one-hot encoding for each defined character ensures control of the input dimensions for the network and thus constrains the RAM usage and with the 140 reading window for a abstract gives a defined input size.

Both RNNs with LSTM-gates and CNNs have shown to work as character-level classification models by Karpathy and Xiang Z. Et al. respectively. [1] [11]

We opt to use the Unicode 5.2.0 encoding format for 100 characters. The 100 encodings selected to include: uppercase letters, lowercase letters, numbers and most common symbols. A *unknown* character was added resulting in a 140x101 input matrix to a character-level model.

3.3. Recurrent neural networks

Recurrent Neural Networks (RNN) were developed to handle sequential data, this is accomplished by introducing a temporal reference in the network, that have a dependence on what previous nodes have seen in time. This is also called model "memory". We assume RNNs should work well with out problem, since text is sequential by nature.

Several specific architecture of RNNs can be considered. The simple RNN (S-RNN) have been shown to work for sequence tagging and language models, but the field have discovered problems with vanishing gradients when applying them to real scenarios. Here the LSTM gates can help by introducing *memory-gates*, A. Karpathy shows in a blog-post how LSTM gates can be utilized as a character-level language model that predicts the next letter in a sequence [1] [7].

3.4. Convolutional neural networks

Convolutional neural networks (CNNs) does not seem straight forward to apply to text. But it have been shown they can be useful as word-level classification models. This is accomplished with the CNNs ability to find strong local association in the input sting. Y. Goldberg gives several examples to working CNNs for text classification in a survey of neural networks for NLP problems [7].

But do CNNs work on a character-level classifier? Xiang Z. et. al. compared a character-level classifier to traditional approaches e.g. word2vec, TF-IDF, n-grams etc. The character-level CNN work without the need of words, they showed text strings can be treated like a regular signal, like images [11].

3.5. Evaluation metrics

For both RNNs and CNNs working as character-level models we want a metric to assess the models.

The network output gives the probability that the abstract belongs to each class. Using those, we define top-k accuracy as *the ratio of times the predicted probability for the correct class is among the top k predicted probabilities of all the existing classes*.

Please note that from the confusion matrices[2] we created, any other evaluation metric can computed later on.

3.5.1. Baselines. Now we present three baselines for top-1 accuracy.

A model that guesses the correct category uniformly at random has 12.5% top-1 accuracy; a model that guesses randomly, but whose distribution of guesses matches the distribution of the categories has a top-1 accuracy of 27.11% and a model that learns to guess always Physics has a top-1 accuracy of 36.04%.

4. Experiments and results

4.1. Preliminary experiments

The very first experiment we ran was based on a basic architecture consisting in just one hidden LSTM layer and a linear layer producing the output probability vector. The aim at the beginning was to tackle the entire complexity of the problem, trying directly to test our first net on the complete classification task of a paper into all the possible eight classes we are interested in. Using just the first 50 characters of the paper abstract, our net was trained using a Cross Entropy Loss function optimized with a classical SGD algorithm. The results we observed were not good at all, and even after trying some hyperparameter tuning on the learning rate value or the batch size dimension, the net was not learning (check Figure 10).

2. All our confusion matrices have the correct classes on the rows and the predicted classes on the columns. Also, the order from top to bottom matches the order from left to right.

This poor result led us to attempt an incremental approach, rather than directly tackling the final task. So we ran a new experiment, in which the same net was trained over just two different categories, namely, Maths and Physics (being the two largest categories in our dataset). Even though the possible content of the abstracts of these two classes of papers is likely to be similar, we were confident in the LSTM ability to still find a way to distinguish them. We were disappointed by the results, as they again showed no trace of learning, as can be seen in the confusion matrices of Figure 11a and 11b.

At this point, we started doubting about the actual correctness of our code implementation. We thought to break this sneaking suspicion with a very simple yet meaningful experiment, in which our net was trained to distinguish between real abstracts and completely random generated text, without real words in it. Gratefully, we could tell from the result in Figure 2 that our net was able to learn this task very efficiently even with a little training time, telling us that we did not have major bugs or information leakages in our code.

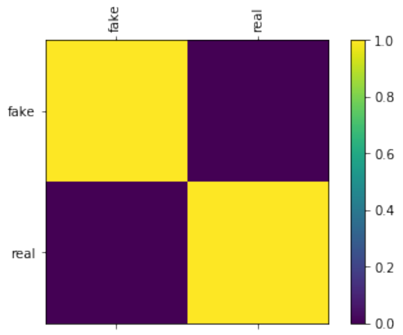


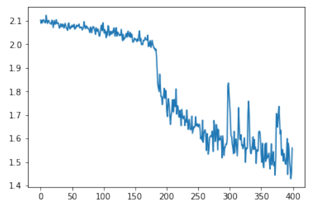
Figure 2: Confusion matrix obtained after training the net with 50 characters of real or fake text

Eventually we were sure that our basis were solid and therefore we could proceed again with further experiments.

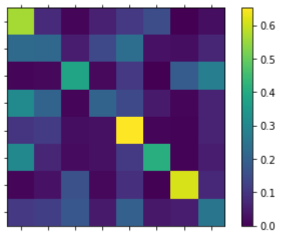
4.2. LSTM continued

Knowing that our net was actually able to learn, at this point we ran a very long experiment in which the same architecture was again trained, this time using the whole length of the abstracts we had. In this way we also checked for the importance of having longer and more structured data to train our net on. The batch size was set to 24 during this, while the learning rate was 0.01, a number that was producing the "more promising" results we had observed so far. This settings produced the learning curve in Figure 3a and the confusion matrix in Figure 3b.

Here, it can be observed that actually even with one layer our RNN was able to get a flavour of the differences that exist between different types of abstracts, though the



(a) Learning curve obtained by training of the net using the whole length of the abstracts



(b) Confusion matrix obtained from the testing of the shallow net on the whole length of the abstracts

Figure 3: The 'long' experiment with the full abstract gives more promising results

accuracy reached (around 41%) is not enough to claim this as our ultimate result.

This result gave us also some hint on the fact that probably our initial choice of considering just the first 50 characters from every abstract was significantly penalizing the performances of the RNN. Moreover, we noticed that almost for half of the training time the performances of the net had not improved over time, even with a much richer dataset. This very long learning time could also have been the cause for the bad results we got in the previous subsection, where it is possible that our net did not even have the time to tune its weights according to the training data.

A new set of experiments was therefore begun, aiming both at investigating the answer to one of our core questions ('How short can we make the passages and still get a reliable classifier?') and at improving the performances by means of using deeper and more structured nets. The new architecture that we tried had 2 LSTM layers, dropout of 0.9 and used the Adam optimizer. Several experiments were run, where the number of characters used was spanning from 50 to 700 (that corresponds more or less to half of the length of most of the abstracts). Some short reading of some abstracts proved us that also for us was very difficult to infer the category of a paper even reading the first 150 characters (cf sections 2 and 6.1), so most likely the bad results in the previous subsection were also due to an excessive restriction of the training data for our model. The vast majority of the results we got with the new net were still not satisfactory, especially for shorter abstract excerpts, but we observed that a certain combination of abstract length (500 characters) and learning rate (0.01) produced a decent result out of the blue (as can be seen in Figure 4), while for example the same learning rate with larger number of characters had poor learning curves.

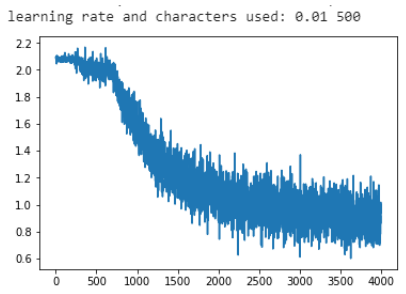


Figure 4: Learning curve obtained with our second net (2 LSTM layers) by reading 500 characters of each abstract.

This very promising result unfortunately was obtained when very little time left before the deadline. Nevertheless, we tried to replicate these results with a deeper RNN network with 10 LSTM layers. Unfortunately we were not able to improve the results we had so far, and the lack of time did not allow us to thoroughly understand why. So the result in Figure 4 remained our best attempt to study the minimum amount of characters needed for an LSTM to correctly classify papers.

4.2.1. Word embedding. In section 3.2.1 we introduced the notion of word embedding, but the vast majority of our experiments used a character one-hot encoding, as explained in 3.2.2. For the sake of completeness we decided to survey the potential of using a word embedding. We used google's pretrained word2vec model [5] to vectorize the words in abstracts and train an LSTM on these vectors as sequential inputs. In particular, we train on 90% of our dataset and test on the remainder, training for 400000 steps with a learning rate of 0.005 and a momentum of 0.9. The test results of this model can be seen in Figure 5

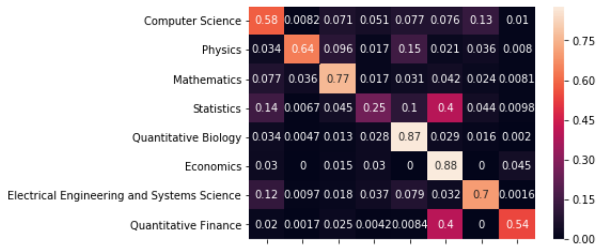


Figure 5: A confusion matrix for the testing of the word2vec based RNN model.

The model got a top-1 accuracy of 65%. Considering that it uses a recently state-of-the-art word embedding model to transform abstract strings into semantically encoded vectors, the performance of this method is quite underwhelming. However, the pre-trained word2vec model did outperform all our LSTM/RNN experiments with the character one-hot encoding and is on par with our CNN model that reads 140 characters (see section 4.3).

4.3. CNN experiments

By the time we had finished the experiments outlined in section 4.1 we hadn't been able to obtain satisfactory results with LSTM/RNN architectures. Because of that, and with [11] as our main reference, we decided to also test with CNN architectures. For that matter, we assembled the architecture of Figure 6 where we fixed the input size at 140 characters. That is, unless explicitly stated, all experiments in this section were performed by reading the first 140 characters of each abstract and by encoding each character in one of 101 classes, as described in section 3.2.2.

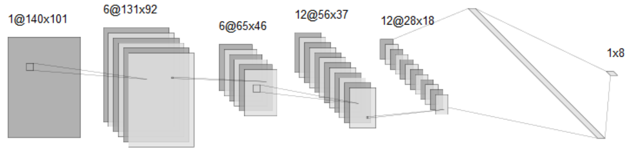


Figure 6: Main CNN architecture used in section 4.3. The fully connected linear layers in the end are three, even though the scheme only shows two.

One cannot say the choice of architecture in Figure 6 was informed and scientific. Rather, we opted for two convolution layers and three linear layers because we hypothesized that would be enough and it still made up for a rather shallow network. The pooling used was 2×2 max-pooling and the non-linearity applied after the convolutions started as the rectified linear unit. At some point we changed to leaky rectified linear units, as we were getting better results with those.

We performed several experiments with the given CNN architecture to find sensible hyperparamenters such as learning rate, training time and batch size, until we got some good results: Figure 7 shows the confusion matrix of a CNN model that achieved 64% top-1 accuracy and 81% top-2 accuracy.

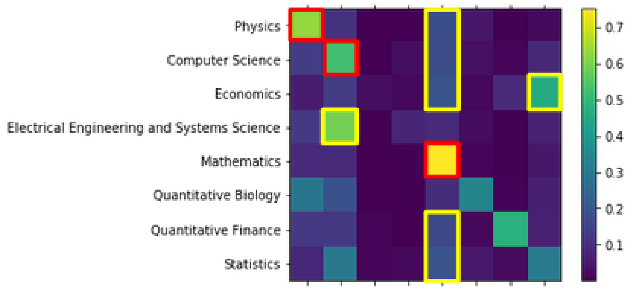


Figure 7: Confusion matrix of a CNN model with 64/81% top-1/2 accuracy. 2 out of every 3 times, the model is able to predict the correct category.

The confusion matrix in 7 also includes some annotations in red and in yellow. The squares marked in red indicate the 3 categories for which the model was most accurate, which coincide with the 3 categories that are more well represented in the dataset. The squares surrounded by yellow contain what we would call *honest mistakes*: systematic mistakes that, from a human perspective, 'make sense'; for example, we can see that on the EESS row there is a green square that matches CS, meaning lots of EESS papers were classified as CS; similarly, on the Economics row we see a green square on the Stats column, meaning a big portion of Economics papers were classified as Statistics papers. We also see that some categories are classified as Mathematics quite frequently.

4.3.1. Data augmentation. At this point we decided to try a slightly different approach with respect to the creation of the training samples, in an attempt to attenuate the fact that our dataset is highly unbalanced (see table 1). In fact, we performed a couple of experiments with the same model and the same character encoding, but instead of extracting the *first* 140 characters of the abstract, we pick a random position in the abstract and extract 140 characters starting from that position. This can be seen as a data augmentation technique, because each paper can now yield $l - 140$ different training samples (although they are not completely independent) if l is the length of its abstract. With this technique, we trained a new CNN model that achieved a top-1 accuracy of 49% and a top-2 accuracy of 71%, as can be seen in Figure 8.

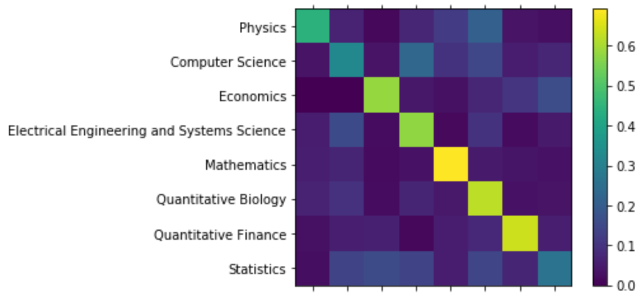


Figure 8: Confusion matrix with the CNN model trained with data augmentation.

The top-1 and top-2 accuracies are worse when compared to the model in Figure 7 but it is of paramount importance to point out that, even though both models trained for the exact same number of iterations and with the exact same batch sizes, this model was exposed to only $\approx 1\%$ of the data available, whereas the better model was exposed to all of the data. So indeed this method of data augmentation seems very promising. A qualitative comparison of Figures 7 and 8 also reveals that for this model, the errors are more evenly spread over the whole matrix and the diagonal seems more well defined. This hints at the fact that the model was still learning how to correctly classify all categories and wasn't overfitting to the 'honest mistakes' mentioned previously.

Two other sensible experiments to assess the impact of having such an imbalanced dataset are outlined in sections 4.3.2 and 4.3.3. These experiments were suggested to us at the poster presentation and for obvious time constraints they were not thorough. We believe that we were able to get some interesting insights nonetheless.

4.3.2. Dropping Economics. Another possible way of fixing the imbalance of the dataset, or at least a way of making it slightly less imbalanced, is by removing the least represented n categories. In our case, with $n = 1$ this amounts to removing the Economics category and checking how the performance of our models behave. The performance gains we managed to obtain are practically negligible, as the best settings for hyperparameters yielded a model with 65/83% top-1/2 accuracy, only a 1% increase in each metric. Refer to figure 12a in the appendix for the model's confusion matrix.

4.3.3. Using the top 3 categories. A symmetrical formulation of the experiment in 4.3.2 would be keeping the n most relevant categories, instead of dropping the n smallest ones. Based on table 1 we opted to keep the $n = 3$ biggest categories, as those have the same order of magnitude of data points. By only having to classify abstracts into Physics, Mathematics and Computer Science we obtained a model with 78/94% top-1/2 accuracy. Please refer to Figure 12b in the appendix for the model's confusion matrix.

4.3.4. Varying input size. To address directly the research topic of how reliable a classifier can be, depending on the number of characters that we feed it, we took the settings of the model in Figure 7 and varied the input size from 50 to 270, inclusive, in steps of 10. We trained all the models for the exact same number of iterations and the train/test data split was the same for all models (i.e. the abstracts used for training were always the same and the abstracts used for testing were always the same). In Figure 9 we can find the top-1 and top-2 accuracies of the models plotted against the input size, together with the baseline of always guessing physics and the top-1 accuracy of the model presented in Figure 7.

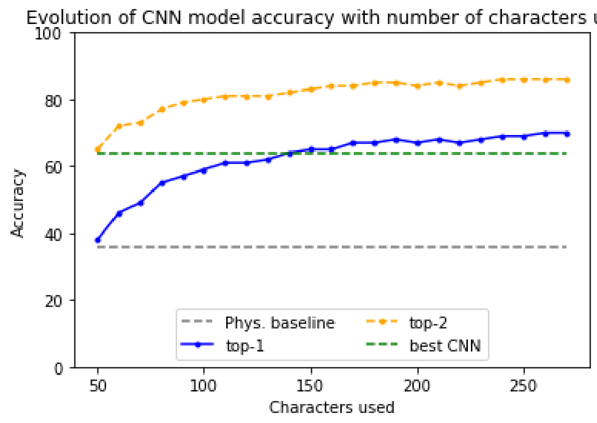


Figure 9: Top-1 and top-2 performances of different CNN models which read different numbers of characters.

From Figure 9 we can see that the performance gains we get by using more characters start to stagnate close to 200 characters, where the performance is around 65/83% for top-1/2 accuracies. The figure also shows that with as little as 80 characters we can get a simple model that achieves 80% top-2 accuracy. Furthermore, we hypothesize if we used the data augmentation technique in section 4.3.1 we could get even better models with very short abstract excerpts.

5. *Research answer* and conclusions

In this section we present our concluding remarks and, in particular, address the research questions mentioned in section 2.

The first couple of remarks we present are already well-understood in the ML/DL community and so they might seem superfluous:

1) Models can be *very* sensitive to hyperparameter settings;
2) Choosing a good model architecture with 'potential' is very important (in particular, see [12]);
3) Models struggle to learn good classifiers for very imbalanced data.

We decided to include these remarks nonetheless because there is a very big difference in reading/being told about something and actually experiencing it. Having struggled directly with all these problems during our project made this a more well-rounded group project.

We also present the following non-trivial findings:

- Despite our remark 2 we also found out that, once a promising architecture was found, it was quite robust to minor changes in its settings; for example, the performance of the CNN architecture we outlined in Figure 6 revealed to be quite robust to changes in the non-linearity used after the convolutions or the exact sizes of the fully connected layers;
- Regarding our research topic of whether we needed a text encoding that preserved semantics, we know we do not. One-hot character encodings can also provide competitive results, even though word embeddings seem to work better 'out-of-the-box', as it can be seen in section 4.2.1;
- Regarding our research topic about the relationship between input size and model performance, section 4.3.4 shows that we can get very reliable models with input sizes for which a human being would struggle to perform on par with the model. Our findings also suggest that the performance gains stagnate as we keep increasing the input (see figure 9) and that the stagnation is below 100% accuracy.
  - We know, however, that using the data augmentation technique in section 4.3.1 would improve the performances, and
  - Papers on arxiv are sometimes cross-posted on different categories and thus it is conceivable that a perfect classifier doesn't even exist because not all papers have a well-defined unique category.

  (the reader is challenged to beat our model in appendix 6.1)

- From the experiments in sections 4.3.2 and 4.3.3 we see that removing some categories makes the classification task easier (but also less interesting at the same time).

All in all, we see that shallow CNN models can make use of a character encoding to classify papers, from their abstracts, without needing long excerpts and achieving impressive performances, when compared to the human baseline. LSTMs need a more careful approach but might also be made to work reasonably.

Finally, we would like to put forward some possible reasons as to why we struggled so much specifically with the RNN/LSTM models, which we still believe could be made to work:

- Character encoding of the abstracts (used in sections 4.1 and 4.2) might be leading to a very sparse representation of the data, and this can be a reason for which our LSTMs were not able to learn so well;
- Considering so many different types of abstracts, written by different people in different fields of human knowledge, leads the dataset to be very varied, not having everywhere the same writing-style as in the Shakespeare text generation via LSTMs described in [1], our main reference as to why we thought this would work; in other words, our task is more complex than the task in [1];
- It is also possible that we focused too much on very shallow nets, mainly at the beginning, in order to try to find some sort of "minimal architecture" for the task, that apparently needed to be more complex than we initially thought.

Acknowledgements

We would like to thank the Deep Learning staff for arranging Google credits to be used in the Google Cloud platform[3] and in particular, we would like to thank the teaching assistant Chia-Lun Yeh for her continuous feedback throughout our whole project and for the technical help she also provided when needed.

We would also like to thank everyone that gave us feedback on our project during the poster session; that feedback turned out to be very valuable and ultimately gave rise to the content in sections 4.3.2 and 4.3.3.

References

[1] A Karpathy. "The unreasonable effectiveness of RNNs". http://karpathy.github.io/2015/05/21/rnn-effectiveness/. [Accessed May 2019].

[2] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, and Y. Chi. "Deep Keyphrase Generation". 2018. *arXiv preprint arXiv:1704.06879v2*.

[3] H. Liang, X. Sun, Y. Sun, and Y. Gao. "Text feature extraction based on deep learning: a review". *EURASIP Journal on Wireless Communications and Networking*, 2017:211.

3. Even though we only used Google Colab as we failed to setup the Google Cloud environment

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space". 2013. *arXiv preprint arXiv:1301.3781v3*.

[5] "word2vec". https://code.google.com/archive/p/word2vec/. [Accessed May 2019].

[6] G. M. Weiss and F. Provost. "The Effect of Class Distribution on Classifier Learning: An Empirical Study". *Technical Report MLTR-43, Dept. of Computer Science, Rutgers Univ.*. 2001.

[7] Goldberg, Y. (2016). *A primer on neural network models for natural language processing*. Journal of Artificial Intelligence Research, 57, 345-420.

[8] Camacho-Collados, J., & Pilehvar, M. T. (2018). *From word to sense embeddings: A survey on vector representations of meaning*. Journal of Artificial Intelligence Research, 63, 743-788.

[9] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). *Distributed representations of words and phrases and their compositionality*. In Advances in neural information processing systems (pp. 3111-3119).

[10] *A brief history of word embeddings (and some clarifications). (n.d.)*. Retrieved July 5, 2019, from https://www.linkedin.com/pulse/brief-history-word-embeddings-some-clarifications-magnus-sahlgren

[11] Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in neural information processing systems (pp. 649-657).

[12] M. Saxe, Andrew & Wei Koh, Pang & Chen, Zhenghao & Bhand, Maneesh & Suresh, Bipin & Y. Ng, Andrew. (2011). On Random Weights and Unsupervised Feature Learning. Proceedings of the 28th International Conference on Machine Learning. 1089-1096.

6. Appendix

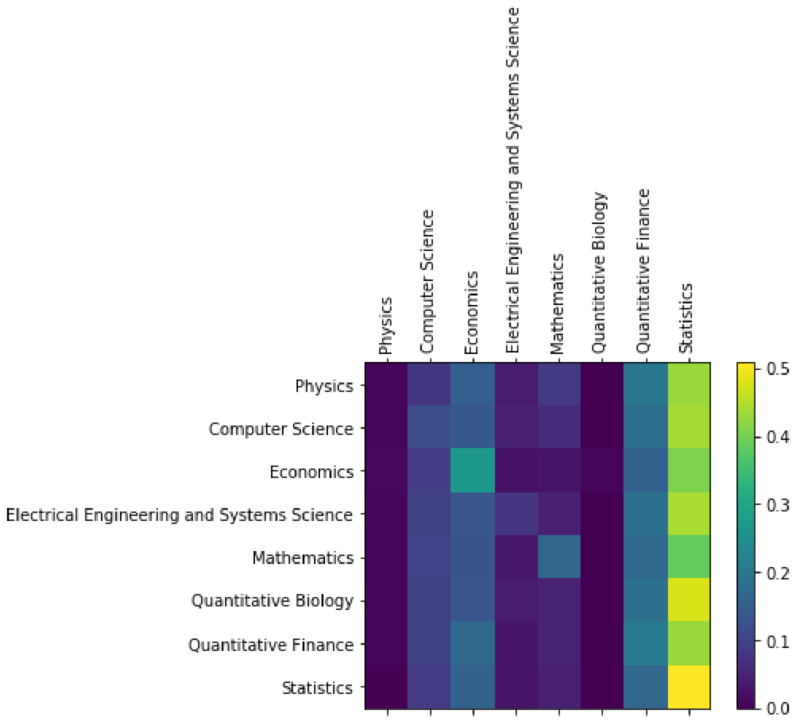
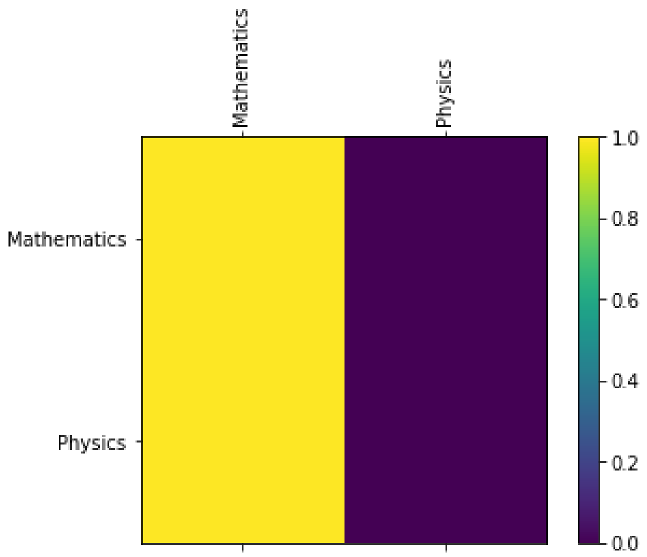
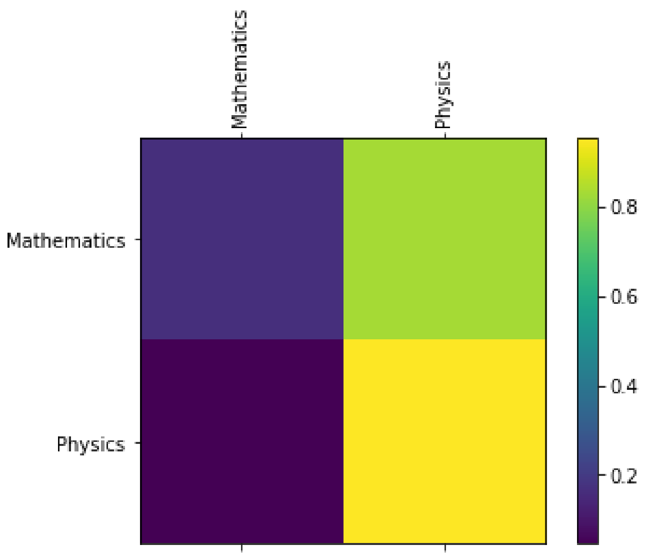


Figure 10: Confusion matrix obtained with our very first experiment, testing the net trained with just the first 50 characters in the paper abstract,using a learning rate of 0.001 and a batch size of 12. On the diagonal we would expect to have much higher values, indicating that the model prediction matched the actual class the papers belong to.

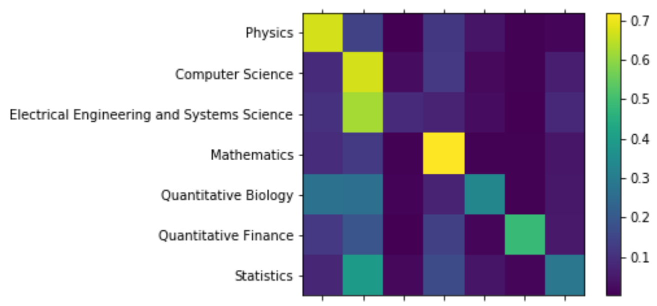


(a) Confusion matrix obtained with our second experiment,where just Math and Physics were considered between all the 8 categories in the dataset. The learning rate used was 0.001

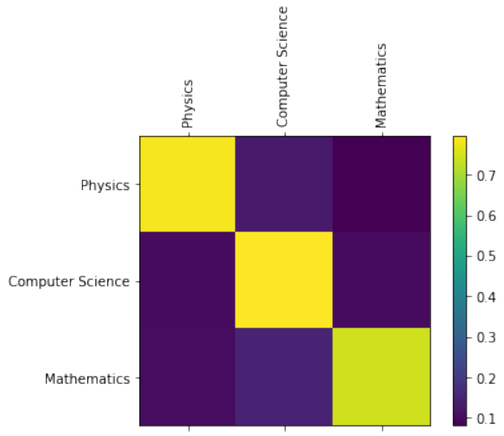


(b) Confusion matrix obtained with our second experiment, with a learning rate of 0.005

Figure 11: Preliminary experiment confusion matrices



(a) Confusion matrix of a model trained on the dataset without Economics papers, as outlined in section 4.3.2. The model achieved a top-1 accuracy of 65% and a top-2 accuracy of 83%.



(b) Confusion matrix of a model trained only on Physics, Mathematics and Computer Science papers, as outlined in section 4.3.3. The model achieved a top-1 accuracy of 78% and top-2 accuracy of 94%.

6.1. Sampling of results for a CNN model reading 140 characters

We will now provide 8 passages of approximately 140 characters. We challenge the reader to categorize each passage; we also present the top-2 predictions of the model and reflect upon its predictions.

Please notice that the following comments are based purely on speculation, from the point of view of a writer with CS and Maths as major background and some familiarity wih Physics, as well as very little knowledge of Economics/Finance. During a poster presentation, the authors were able to witness different people guess correctly passages that seemed impossible to guess and people failing to guess passages that seemed obvious; even if making use of sound reasoning, the ability to guess correctly is limited by the knowledge each person has of each area.

6.1.1. "SAP is the market leader in enterprise software offering an end-to-end suite of applications and services to enable their customers worldwide [...]" (141). The model predicts (CS, Stats) and this is, in fact, a CS paper. A human could arguably be deceived by the reference to SAP and try to guess Quant. Finance or Economics. Presumably the model learned to associate words like 'software' to the CS category. https://arxiv.org/abs/1906.07154

6.1.2. "It is common practice in using regression type models for inferring causal effects, that inferring the correct causal relationship requires [...]" (139). The model predicts (Stats, CS) and this is, in fact, a Stats paper. This is an "easy" passage given the appearance of 'regression' and 'causal effects'. https://arxiv.org/abs/1906.071360

6.1.3. "The research presented in this article provides an alternative option pricing approach for a class of rough fractional stochastic volatility [...]" (140). The model predicts (Maths, CS) while this is a Quant. Finance paper. The model could've been inclined to guess Maths because of the references to 'fractional' and 'stochastic' and a human could associate 'pricing' to Economics. This paper is actually cross-posted under Probability, Maths in arxiv. https://arxiv.org/abs/1906.07101

6.1.4. "Sexual reproduction is not always synonymous with the existence of two morphologically different sexes; isogamous species produce sex cells [...]" (139). The model predicts (CS, Phys.) while this is a Quant. Biology paper. A human would easily guess this category from the references to 'cells' and 'sexual reproduction', for example. The predictions are for two of the three biggest classes available and there are no obvious references in the abstract that hint at any of those predictions, hence it looks like the model is trying to guess randomly (and optimally) because it cannot generalize to this abstract. https://arxiv.org/abs/1906.07117

6.1.5. "To address the issue that Lagrangian dual function based algorithms cannot guarantee convergence and global optimality for decentralized [...]" (136). The model predicts (Stats, CS) while this is a EESS paper. Given the references to 'Lagrangian' + 'global optimality' and 'algorithms', both predictions make sense from a human perspective. https://arxiv.org/abs/1906.07126

6.1.6. “This paper introduces a modification of n-dimensional Hopf-Cole transformation to the n-dimensional Burgers’ system. We obtain the n-dimensional [...]” (144). The model predicts (Phys., Maths) and this is a Maths paper. It is debatable that this passage hints at a n-dimensional system of differential equations in a physical setting; hence the possible confusion. https://arxiv.org/abs/1906.07150

6.1.7. “We study a pure-exchange incomplete markets model with heterogeneous agents. In each period, the agents choose how much to save and which [...]” (137). The model predicts (Phys., Quant. Finance) when this is, in fact, an Economics paper. A possible confusion with physics could come from ‘heterogeneous’. References to ‘markets’ would guide a human to guess Economics or Quant. Finance. https://arxiv.org/abs/1906.06810

6.1.8. “Pulsars typically exhibit radio emission in the form of narrow pulses originated from confined regions of their magnetospheres. A potential [...]” (139). The model predicts (Phys., Quant. Biology) and this is a Physics paper. References to ‘pulsars’, ‘radio’, ‘pulses’ make this an easy guess for humans and for the model. https://arxiv.org/abs/1906.07156

All in all, the model guessed $3/8 = 37.5\%$ on the first guess and $4/8 = 50\%$ on the top-2 guesses for this sample of 8 new abstracts. Of course this is not alarming because the accuracy measures are calculated with more than 50.000 abstracts.