Unsupervised pretraining for text classification using siamese transfer learning

Notebook for PAN at CLEF 2019

Maximilian Bryan, J. Nathanael Philipp

Universität Leipzig
bryan@informatik.uni-leipzig.de, jonas_nathanael.philipp@uni-leipzig.de

Abstract When training neural networks, huge amounts of training data typically lead to better results. When only a small amount of training data is available, it has been proven useful to initialize a network with pretrained layers. For NLP tasks, networks are usually only given pretrained word embeddings, the rest of the network is not pretrained since pretraining recurrent networks for NLP tasks is difficult. In this article, we present a siamese architecture for pretraining recurrent networks on textual data. The network has to map pairs of sentences onto a vector representation. When a sentence pair is appearing coherently in our corpus, the vector representations should be similar, if not, the representations should be dissimilar. After having pretrained that network, we enhance it and train it on a smaller dataset in order to have it classify textual data. We show that using this kind of approach for pretraining results in better results comparing to doing no pretraining or only using pretrained embeddings when doing text classification for a task with only a small amount of training data. For evaluation, we are using the bots and gender profiling dataset provided by PAN 2019.

1 Introduction

When training a neural network for a given task, training data is needed. With this data, the network adjusts its weights in order to correctly classify input data onto given labels presented with the training data. The more variance there is in the input data, the better the network performs on unseen data. Thus, the more training data is available, the better a network is performing [13]. On the other hand when there is not much data available, a neural networks is running the risk of overfitting the training data since only patterns are learned that can be seen in the training data.

In this article, we present an architecture for pretraining neural networks on textual data, that not only pretrains word embeddings but is a bigger network that learns sentence representations. In section 3 we present a siamese architecture to pretrain a neural network on a huge text corpus. To show that this pretraining leads to better results when transferring the network to a specific task, we enhance the presented architecture and continue training it on a smaller dataset. The smaller dataset will be taken from the bots and gender profiling task of PAN 2019 [6,23,24,22]. In section 4, we compare this pretrained network with a not-pretrained network.

2 Related Work

In order to circumvent the problem of having little training data for a specific task, a neural network can be trained on other, related data before being trained on the data for the desired use case. This approach is known as transfer learning [20] and is commonly used in the field of computer vision by using the pretrained *ImageNet* [12], which is trained on 1.2 million images, on tasks like image classification [9], object detection [10], image segmentation [7] or others. In all use cases, the network is performing better because it has been pretrained on a large dataset of images.

Pretraining neural networks for NLP tasks is harder because of two reasons: on one hand, it is very easy to gather text data, but it is much more difficult to get a labeled corpus of humongous size. On the other hand, text data is unsuited for use of traditional pretraining architectures like autoencoders [27] for several reasons. When pretraining unsupervised using an autoencoder, the neural network has to recreate the given input data. Now, when the input data is a sequence of dictionary entries, the network has to contain a layer that is mapping onto a dictionary in order to e.g. predict probabilities of tokens being present. Given that dictionaries typically contain several hundreds of thousands of tokens, that single layer contains so many weights that it becomes technically hard to train such a network. Also, when using an autoencoder as a pretraining step, one is usually only interested in the encoding part of the network, but the decoding part still has to be trained which results in bigger networks than actually needed. Another problem arises when considering that the fact that text data is sequential. This makes autoencoders, which work great for images, unusable. One could think of a recurrent architecture that maps a sequence onto itself, but when using that kind of architecture, the network just has to map each token onto itself and does not need to consider contextual information. To circumvent this problems, architectures have been created which split the network into an encoding and decoding part by mapping the sequence onto one vector representation and then recreating the sequence using that single vector [25,3].

To avoid the aforementioned problems when pretraining sequential networks in the context of NLP, networks most often contain a pretrained input part, which maps words to a vector representation, often called embedding and created using tools like *Word2Vec* [18], *GloVe* [21] or *context2vec* [17]. For example, [16] have created an architecture for question answering using pretrained word vectors using Glove , [2] created a neural network for natural language inference, in which they used pre-trained word vectors, [26] used a pre-trained embedding for creating a neural network to do semantic role labeling.

3 Siamese Architecture

A siamese architecture belongs to the group of unsupervised learning methods. It does not aim to perform a classification like with supervised approaches but tries to find patterns in data, so that the given input data can be mapped onto a vector representation.

This approach is similar to Restricted Boltzman Machines [14] or Autoencoders [27]. A siamese architecture is similar to an autoencoder's architecture, whereas it only contains the first part which is the encoding part. Due to the lacking of a decoding part, the siamese architecture cannot be trained to recreate the input data but is instead trained using data pairs. The network is given two entries of the training set and has to map both inputs onto the same vector representation as long as both entries are considered to be similar (positive pair). The vector similarity can then be calculated several ways, e.g. using cosine similarity, euclidian similarity or the Jaccard index. It is important to give the network an input pair of which the both entries are considered being not similar (negative pair). Otherwise, the network would convert to the trivial solution of giving all input data the same vector representation. Using the positive/negative pair approach, the network has to find patterns in the input data so that positive input pairs with similar patterns result in a similar network representation and conversely negative pairs result in different vector representations.

To our knowledge, the first application of such architecture was created by [1] for verifying signatures. A neural network was created that mapped the input image onto a vector representation. Then, two signature's cosine similarity had to be 1 when the signatures where from the same person. Another siamese approach was created by [4]: They created a convolutional neural network with a linear output. During training, the network was trained to give face images from the same person a cosine similarity of 1. When the images came from two different persons, the cosine similarity should be 0. Other siamise architectures have been used for text use cases. Putting text character-wise into LSTMs was done by [19]. They used job descriptions, which had been labeled by similarity previously. The neural network then had to create a vector representation, which then has been used with cosine distance to create the similarity between input data.

Network Structure When training a neural network to learn semantical and structural patterns from sentences, we propose to use a recurrent network. The recurrent network is fed with a sequence of word vectors. To allow for out-of-vocabulary tokens, we add a second input per time step, which is a character ngram representation of the current token. That way, word vectors are trained together with the character of the corresponding token. When the token is not in our dictionary, we use a zero word vector and weigh the character ngram representation with a factor of 2. As a type of recurrent layer, we advocate for using a GRU [5]. As a sentence's vector representation, we are using the last state of the recurrent network. Using a GRU over a simple RNN has the advantage that a GRU solves the vanishing gradient problem of plain RNN layers. When comparing an GRU with the also commonly used LSTM layer [15], there is a problem with the LSTM's hidden state: The LSTM cell is designed such that it contains a state which is protected and will only be output when the corresponding gates *opens*. At the same time, an freely accessible hidden state is passed over each time step. That means, there are two vector representations of each LSTM cell at each time step. Since we do not want to decide which state to take and we neither want to combine two vectors which different purpose, we decide to use the GRU with its single vector representation. Also, since the GRU has less gates and thus less weights to train, the overall training speed increases compared to using an LSTM. The explained structure can be seen in figure 3.

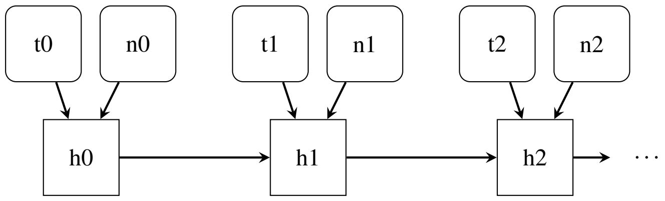

Figure 1. Architecture for multi-class visual attention. Tokens are given into a recurrent network as pairs of word vectors and character ngram representations.

Training Data For unsupervised training of a neural network on sequential textual data, we are using a huge collection of crawled news articles [11]. When comparing news articles to books or tweets, they have the advantage that each article contains a rather short set of sentences of which we know that they cover the same topic. When training the network using the positive negative approach, we create positive pairs by choosing random coherent sentence pairs from the same news article. Negative pairs consist of random sentences from the whole corpus. The network has to learn to map sentences onto a vector representation whereas it has to learn which sentences are likely to be paired with a given sentence, which means that features from possibly coherent sentences have to be contained in a sentence's vector representation. At the same time, all sentences cannot be mapped onto the same vector representation because of the negative pairs. The effect is that sentences like *Python is a nice programming language.* and *I like working with it.* probably appear more often as a positive pair than a negative pair, thus the network has to learn to give both sentences a similar vector representation. The fact that sentences pairs are chosen as positive pairs but can also be shown as negative pairs is called *weak supervision* ([8] among others).

4 Evaluation

In this section, we want to evaluate our pretraining approach. We are using a rather small dataset provided by *PAN @ CLEF 2019* [6]. The dataset consists of tweets of different authors [24]. The authors have to be categorized whether they are a bot or a human being, and if the latter, whether the author is male or female. For that task, tweets of 2880 authors are given as training data. When comparing that amount of authors with the number of users a service like Twitter sees daily, it is incredibly small. With that small amount of data, the problem now is that typical tweet patterns or even linguistic patterns may not be presented often enough in the training data that the network can robustly learn to recognize these patterns. Also, some patterns can occur above-average in the training data, so that the network could fit itself to the training data, which then is disadvantageously when using it on data unseen during the test phase.

The architecture used for this challenge will be the one described in section 3. Since the network only returns a vector representation of a single sentence and no classification for an author, we enhace it with a recurrent layer and two small output layers. The first has two outputs which indicates the information *bot* or *no bot*, the second additional layer has three outputs and indicates *bot*, *male* or *female*. Both new output layers are activated using softmax. The architecture can be seen in figure 4.

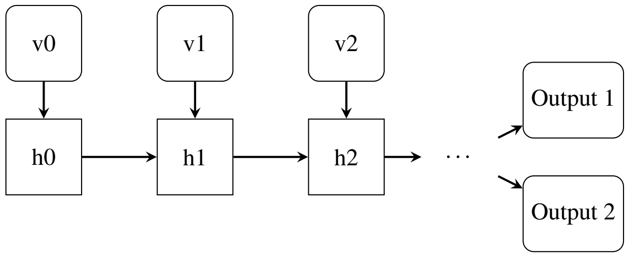

Figure 2. Enhanced architecture for taking part in the author-profiling-challenge. The input into the recurrent layer is a tweet's vector representation created by the network seen in figure 3.

	bot / no-bot		bot / male / female	
	English	Spanish	English	Spanish
not-pretrained	0.6514	0.5708	0.6758	0.6525
pretrained embeddings	0.7303	0.6101	0.6928	0.6897
pretrained siamese	**0.8597**	**0.7862**	**0.7494**	**0.7169**

Table 1. Comparison of network accuracy for differently pretrained networks on the challenge's publicly available test data.

We are using three differently trained versions of that architecture for comparison which are differing in the parts which have been pretrained. The training for fine-tuning the data will be done on the data provided for the challenge. The testing will be done on the publicly available test data. The first version of the architecture won't be pretrained at all, i.e. it does not get pretrained word embedding or a pretrained recurrent layer. The second version will only be trained on the challenge data, but will receive pretrained word embeddings. This approach is similar to other previously mentioned related projects, which create a neural network for an NLP task and provide pretrained word embeddings. The third version will be pretrained using the siamese approach described in section 3. After pretraining, the architecture will be enhanced and trained on the challenge's training data. Additionally, each approach for the three described versions is done for English data and for Spanish data. As can be seen in table 1, the presented architecture's performance get significantly better the more pretraining has been done.

5 Conclusion

In this article, we tackled the problem of pretraining recurrent architectures on NLP data by transfering the siamese architecture to textual data. We are training a network using positive/negative sentence pairs with weak supervision. The sentence pairs are chosen from a corpus of news articles. When sentences are appearing next to each other in an article, they are considered being a positive pair, otherwise they are a negative pair. This pretrained network was then enhanced and trained on a smaller dataset to have it classify author's by their tweets. We showed that we get significantly better results than when no pretraining is performed or when just pretrained word embeddings are used.

References

1. Bromley, J., Guyon, I., Lecun, Y., Säckinger, E., Shah, R.: Signature verification using a siamese time delay neural network. vol. 7, pp. 737–744 (01 1993)
2. Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H.: Enhancing and combining sequential and tree LSTM for natural language inference. CoRR abs/1609.06038 (2016), http://arxiv.org/abs/1609.06038
3. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR abs/1406.1078 (2014), http://arxiv.org/abs/1406.1078
4. Chopra, S., Hadsell, R., Lecun, Y.: Learning a similarity metric discriminatively, with application to face verification. vol. 1, pp. 539– 546 vol. 1 (07 2005)
5. Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR abs/1412.3555 (2014), http://arxiv.org/abs/1412.3555
6. Daelemans, W., Kestemont, M., Manjavancas, E., Potthast, M., Rangel, F., Rosso, P., Specht, G., Stamatatos, E., Stein, B., Tschuggnall, M., Wiegmann, M., Zangerle, E.: Overview of PAN 2019: Author Profiling, Celebrity Profiling, Cross-domain Authorship Attribution and Style Change Detection. In: Crestani, F., Braschler, M., Savoy, J., Rauber, A., Müller, H., Losada, D., Heinatz, G., Cappellato, L., Ferro, N. (eds.) Proceedings of the Tenth International Conference of the CLEF Association (CLEF 2019). Springer (Sep 2019)
7. Dai, J., He, K., Sun, J.: Instance-aware semantic segmentation via multi-task network cascades. CoRR abs/1512.04412 (2015), http://arxiv.org/abs/1512.04412
8. Dehghani, M., Zamani, H., Severyn, A., Kamps, J., Croft, W.B.: Neural ranking models with weak supervision. CoRR abs/1704.08803 (2017), http://arxiv.org/abs/1704.08803
9. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: Xing, E.P., Jebara, T. (eds.) Proceedings of the 31st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 32, pp. 647–655. PMLR, Bejing, China (22–24 Jun 2014), http://proceedings.mlr.press/v32/donahue14.html
10. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. CoRR abs/1311.2524 (2013), http://arxiv.org/abs/1311.2524
11. Goldhahn, D., Eckart, T., Quasthoff, U.: Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In: In Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12 (2012)

12. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
13. Hestness, J., Narang, S., Ardalani, N., Diamos, G.F., Jun, H., Kianinejad, H., Patwary, M.M.A., Yang, Y., Zhou, Y.: Deep learning scaling is predictable, empirically. CoRR abs/1712.00409 (2017), http://arxiv.org/abs/1712.00409
14. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006), https://science.sciencemag.org/content/313/5786/504
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9, 1735–80 (12 1997)
16. Liu, X., Shen, Y., Duh, K., Gao, J.: Stochastic answer networks for machine reading comprehension. CoRR abs/1712.03556 (2017), http://arxiv.org/abs/1712.03556
17. Melamud, O., Goldberger, J., Dagan, I.: context2vec: Learning generic context embedding with bidirectional LSTM. In: Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning. pp. 51–61. Association for Computational Linguistics, Berlin, Germany (Aug 2016), https://www.aclweb.org/anthology/K16-1006
18. Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient estimation of word representations in vector space (2013), http://arxiv.org/abs/1301.3781
19. Neculoiu, P., Versteegh, M., Rotaru, M.: Learning text similarity with siamese recurrent networks (01 2016)
20. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. on Knowl. and Data Eng. 22(10), 1345–1359 (2010), http://dx.doi.org/10.1109/TKDE.2009.191
21. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP. vol. 14, pp. 1532–1543 (2014), https://nlp.stanford.edu/pubs/glove.pdf
22. Potthast, M., Gollub, T., Wiegmann, M., Stein, B.: TIRA Integrated Research Architecture. In: Ferro, N., Peters, C. (eds.) Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF. Springer (2019)
23. Rangel, F., Franco-Salvador, M., Rosso, P.: A low dimensionality representation for language variety identification. In: Gelbukh, A. (ed.) Computational Linguistics and Intelligent Text Processing. pp. 156–169. Springer International Publishing, Cham (2018)
24. Rangel, F., Rosso, P.: Overview of the 7th Author Profiling Task at PAN 2019: Bots and Gender Profiling. In: Cappellato, L., Ferro, N., Losada, D., Müller, H. (eds.) CLEF 2019 Labs and Workshops, Notebook Papers. CEUR-WS.org (Sep 2019)
25. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. CoRR abs/1409.3215 (2014), http://arxiv.org/abs/1409.3215
26. Tan, Z., Wang, M., Xie, J., Chen, Y., Shi, X.: Deep semantic role labeling with self-attention. CoRR abs/1712.01586 (2017), http://arxiv.org/abs/1712.01586
27. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders (2008)