Modular Deep Encoder-Decoders

Group 40: Arnold, Gurunat, Huang, & Qu

April 28, 2016

Abstract

In this short paper, we propose a Modular Deep Encoder-Decoder network (MDED) for text classification. The basic idea behind MDED is to extend a pre-trained model, which consists of a set of deep encoder-decoders, with an additional source of data by adding a new encoder to the network. MDED has been applied to text classification on SaudiNewsNet. A modular Long Short Term Memory (LSTM) network is built. Experimental results show that our algorithm can efficiently combine new source of data and pre-trained model. (show some summaries of experimental results here)

I Introduction

In the past decade, deep learning has been successfully applied to natural language processing (NLP). It has been shown that end-to-end trained deep learning algorithms can outperform traditional algorithms on various NLP tasks.

However, the training process of deep learning algorithms is usually very time-consuming. And it is inefficient to update a model and re-train it from scratch everytime there is a new data source coming. To alleviate this problem, a novel Modular Deep Encoder-Decoder network (MDED) is proposed in this paper. The basic idea behind MDED is to extend a pre-trained model, which consists of a set of deep encoder-decoders, with an additional source of data by adding a new encoder to the network.

More formally, we want to jointly learn a set of *encoder* functions $\{E_i\}_0^N$ mapping samples $x \sim \chi_i$ from a set of data distributions $\{\chi_i\}_0^N$ to a fixed-sized vector embedding V.

$$\forall i \in [0; N] : E_i(x) : \chi_i \to V_i \in \Re^M$$

$$V = \sum_i^N V_i$$

The embedding V is then fed into a decoder function D which in the case of classification learns a mapping from the vector space of V to the label space L.

$$D(v) : \Re^M \to L \in \Re^D$$

Finally, we want to extend the set of encoders by learning a new encoder E_{N+1} which handles samples from a different dataset χ_{N+1}, without retraining our trained decoders and encoders.

$$E_{N+1}(x) : \chi_{N+1} \rightarrow V_{N+1} \in \Re^M$$

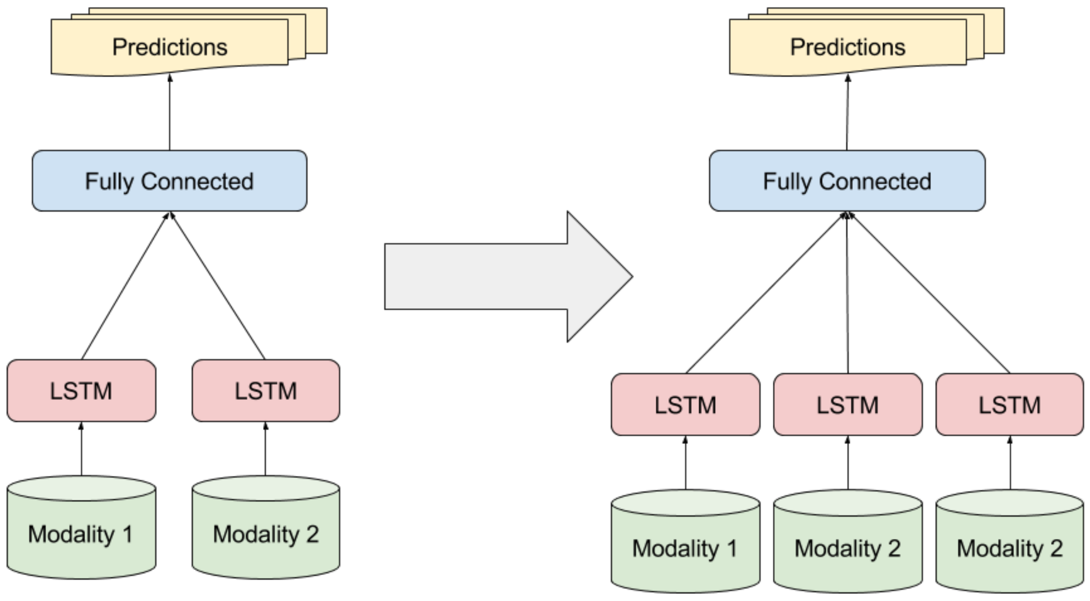

Figure 1: Multi-modal Transfer Learning

Although we could have used any kind of mapping, we chose to use deep learning algorithms, as they easily learn hierarchical representations and have been known to highly outperform other statistical techniques on natural language tasks. Learning embeddings has been previously done by [1], although our proposed contribution is slightly different. [1] used deep autoencoders to obtain a better initialization of the parameters of their model. Our approach instead is much closer in spirit to the work of Sutskever [2] and Vinyals [3]. They both use encoders on a sequence of data to generate a *thought-vector* which will then be decoded in the desired terget sequence. In some sense, our proposal adds the transfer learning component to their contribution. This approach is also similar to the work of Karpathy & al [4] where they mapped images to their captions with embeddings.

II Method

Materials

We used the freely available dataset from `https://github.com/ParallelMazen/SaudiNewsNet` The dataset contains a total of 31,030 Arabic newspaper articles, with a total number of 8,758,976 words.

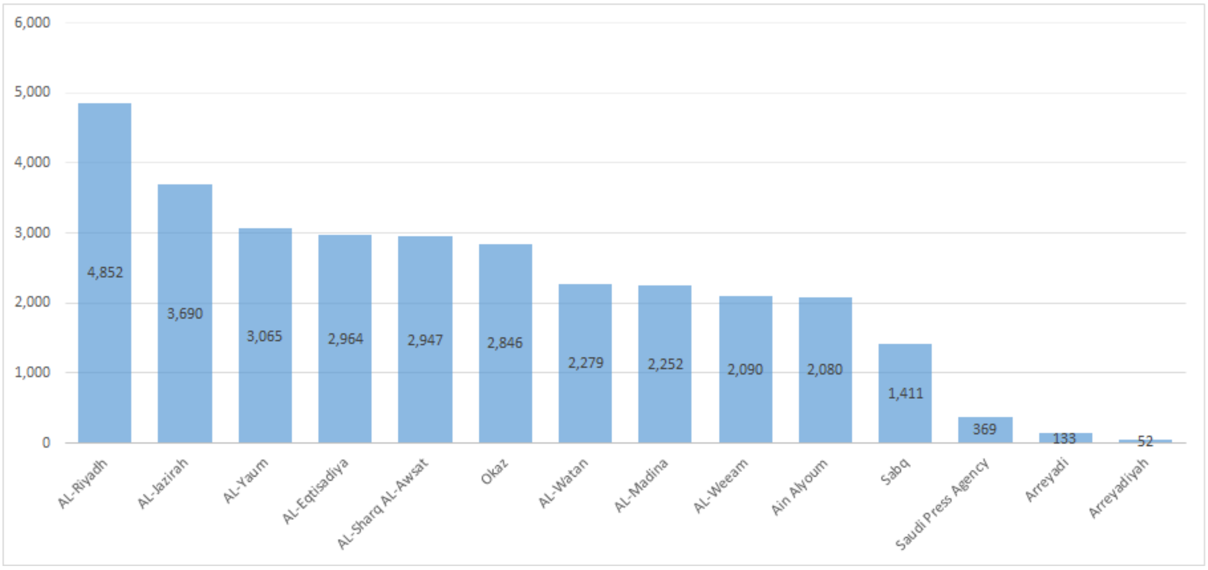

Figure 2: Figure shows article-newspaper distribution

Article objects are reprensented in json format, with UTF-8 encoding. We wrote a script to download the data from github repo, unzip each file, and read them in as json objects. Each json object contains the flollowing fields: source: A string identified of the newspaper from which the article was extracted. It can have one of the following values:

- source: A string identifier of the newspaper from which the article was extracted.
- url: The full URL from which the article was extracted.
- date_extracted: The timestamp of the date on which the article was extracted. It has the format YYYY-MM-DD hh:mm:ss.
- title: The title of the article.
- author: The author of the article.
- content: The content of the article.

Procedure

Data Pre-processing and Features

Firstly, the UTF-8 encoded data was checked for trailing spaces for removal. Next, the data was filtered by removing punctuation symbols, retaining only a subset consisting of {, . ! ( ) ?}. The data was then split into words which are used as tokens for building the vocabulary and finally obtaining a feature representation for training the neural network. A vocabulary is constructed by assigning unique integer word-ids to each unique word appearing in the corpora. The punctuations are treated as a separate token and contributes to the vocabulary. The url, date and title fields of the corpora were discarded since they were irrelevant for our purpose.

After building the vocabulary for the corpora, the original sentences of the corpora are mapped on a word-basis to obtain a series of integers representing the sentence equivalent which form the features to be input to the embedding layer of the neural network. Alternatively, the integer mapping could be replaced with one-hot sparse encoding of each word constituting the sentence. The features are shuffled and split for training and testing. For efficient data storage and retrieval the features are compressed to HDF5 file format.

Embedding Layer

An embedding layer is used to compute a word embedding $W : words \mapsto \mathbb{R}^n$ which is a function mapping words of a language to a high dimensional vectors which are continuous, distributed representation of the vocabulary words with good semantic properties. The hope of such a representation is to project similar words to similar regions in the hyper-space from a semantic point of view.

We tried two approaches when building the vector embedding from multiple modalities. The first one was to vertically stack the different embeddings from the different encoders and learn the actual embedding from a linear combination from this higher dimensional vector. This led to poor results and we instead decided to experiment with both the summing over every embedding, as well as taking their mean in each dimension.

Discriminative Model

We construct two baseline systems, one to classify the journals based on the content of the articles and the second on their authors. Identical architectures are used for both the baseline systems. The baseline system is a recurrent neural network with one hidden Long Short Term Memory (LSTM) layer succeeding the embedding layer. The dimension of both the embedding layer and the LSTM was fixed to 128. The output layer is a softmax layer with 14 outputs, one per class. The model minimize a cross-entropy loss using adaptive gradient descent. [5]

Evaluation

The evaluation procedure for all experiments is based on the classification accuracy of each statistical model on a held-out validation set.

Before any learning happens, we begin by partitioning the dataset in two subsets: training and test. This split is achieved by first loading the entire dataset in memory and assigning the first 80% examples to the validation set and the remaining 20% to the test set. Doing so, we increase our confidence in the resulting accuracy. In preliminary experiments we observed that the majority of the models were highly overfitting the training set. In order to find better hyperparameters, we thus further partitioned the training set into a development and a validation set. (again an 80-20 split) Once the recall-variance trade-off reached a satisfactory level on the development and validation set, we train all models using the same hyperparameters on the entire (identical) training set, and evaluate them on the test set.

Table 1: Classification performance of single LSTM network

Training data	Title	Author	Content
Training accuracy			
Testing accuracy			

Figure 3: Training loss and testing loss of single LSTM networks.

Table 2: Classification performance of modular LSTM networks

Base source	Title		Content		Author	
New source	Content	Author	Title	Author	Content	Title
Training acc						
Tesging acc						

The training for evaluation is performed as follows. We begin by training a single encoder and the decoder on a single modality. (eg, the content of the articles) We measure the obtained accuracy on the test set and serialize the model's parameters on disk. In the second step, we recreate the previously trained encoder and decoder, and initialize them with their respective serialized parameters. We also instantiate a new encoder architecturally identical to the first one but with untrained parameters. Finally, we load both the previous and the new modality from the dataset and their corresponding labels, before proceeding to the training steps as described in the previous sections.

III Results

Single LSTM network

First, three single LSTM networks are trained for text classification, where title, author and content values are used for training data, respectively. The classification performance of them are reported in Table.1. From Table.1 we can see that xxx model has the best classification performance among these three models.

Moreover, the training behavior of these three models are presented in Fig.3.

Modular LSTM network

In oder to further improve the classification performance, new sources of training data are added to the base single LSTM network. For each single LSTM network, two other data source are added to it, respectively. For example, for content LSTM network, author of the articles and title of the articles are added to it as an additional source of training data. Technically, a new encoder is added to the original LSTM network. Performance of these modular LSTM networks are reported in Table.2.

Moreover, the training behavior of them are presented in Fig.4.

Figure 4: Training loss and testing loss of modular LSTM networks.

IV Discussion

References

[1] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning: transfer learning with deep autoencoders," in *Proceedings of the 24th International Conference on Artificial Intelligence*, pp. 4119–4125, AAAI Press, 2015.

[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[3] O. Vinyals, Ł. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, "Grammar as a foreign language," in *Advances in Neural Information Processing Systems*, pp. 2755–2763, 2015.

[4] A. Karpathy, A. Joulin, and F. F. F. Li, "Deep fragment embeddings for bidirectional image sentence mapping," in *Advances in neural information processing systems*, pp. 1889–1897, 2014.

[5] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.