Modular Deep Encoder-Decoders

Group 40: Arnold, Gurunat, Huang, & Qu

April 27, 2016

Abstract

In this short paper we propose an approach to transfer learning using rich vector embeddings. The suggested technique can be applied to any supervised task, and it handles multiple sources and changing sources of data without the need for retraining. To verify our ideas, we apply our ideas to the task of text-classification.

I Introduction

Our goal is to generate rich vector embeddings from articles to classify them into predefined categories. Then, we want to extend our model with an additional source of data: the title of the videos. To handle this new knowledge source without retraining our previous model, we suggest to generate a new embedding that will be used to modify the original one. This combined embedding will then be used for the classification task.

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.

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

$$V = \sum_i^N V_i$$

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.

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

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.

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

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 [3] and Vinyals [4]. 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 [5] 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 title, author, date, url and content in each entry. 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. The function can also give out content by key values such as title, author and etc.

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 $W : \textit{words} \mapsto \mathbb{R}^n$ 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 hyperspace from a semantic point of view.

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. [6]

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.

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.

Paragraph: By using a standardized ds, we can fairly compare models...

III Results

Needs results for:

1. single model on authors 2. single model on content 3. single model on titles 4. multimodal on authors-content 5. multimodal on title-content 6. ? multimodal author-titles ?

Baseline

Proposed System

IV Discussion

V References

Part of the relevant literature review. More literature was involved for the deep learning part.

1. Supervised Representation Learning: Transfer Learning with Deep Autoencoders, Zhuang & al, http://ijcai.org/Proceedings/15/Papers/578.pdf

2. Transfer Learning via Dimensionality Reduction, Pan & al, https://www.cse.ust.hk/~jamesk/papers/aaai08.pdf

3. Sequence to Sequence Learning with Neural Networks, Sutskever & al, http://arxiv.org/abs/1409.3215

4. Grammar as a Foreign Language, Vinyals & al, http://arxiv.org/abs/1412.7449

5. Deep Fragment Embeddings for Bidirectional Image Sentence Mapping, Karpathy & al, https://cs.stanford.edu/people/karpathy/nips2014.pdf

6. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, Ducih & al, http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf