Learning Collective Word and Document Embeddings for Document Categorization

Nitish Gupta

March 2015

1 Introduction

With the increasing growth of textual information on the Web, it has become important to be able to easily predict the topics present in the documents. Though a number of successful approaches (cite related work to document classification here.) have looked at this problem, these methods are not always best suited for multi-class document classification. Also, most of the methods represent documents using a one-hot bag-of-words kind of a feature or go to the extent of representating the documents with their corresponding tf-idf vectors. Recently, distributed word embeddings learnt using probabilistic neural language model (cite bengio) or various other methods (cite others, specifically mikolov, collobart,) have shown to learn vector representations (embeddings) for words that capture the semantic similarity between words. Though, such word embeddings are powerful in capturing word-level semantic closeness, it is not clear how embeddings for documents can be learnt that can be used to predict the topics contained in the document.

The aim of this project is to build a system that is able to jointly learn word and document embeddings to aid the task of multi-class document categorization while at the same time ensuring that the word embeddings learnt capture the semantic content of the words.

2 Methodology

The task of multi-class document classification is broken into 2 steps. Firstly, learning meaningful word and document embeddings for all the documents and words contained in them, secondly using the learnt document embeddings to learn category embeddings using the training data which can then be used to predict the categories that a particular document belongs.

Our model, of learning word and document embeddings, is inspired by the Continuous Bag-of-Words model of (cite CBOW Mikolov) in which, given the embeddings of the words in a context, the aim is to predict the middle word in the context. The CBOW model also looks at the words ahead of the word it wants to predict and can do so beause the aim is not to build a language model, but a system that can predict the centre word in a context. In (cite paragraph vector Mikolov), they extend their previous CBOW model to incorporate a separate paragraph/document embedding along with the context words to improve the prediction of the middle word by incorporating knowledge of the document in terms of its embedding. We hypothesize that document embeddings learnt in such manner should be able to catpture the semantic content and different topics present in the document through the different factors of the embeddings and should be able to perform better at the task of multi-class document classification than a model that has no information about the content of the documents or a simple bag-of-words model. Our model, though very similar to (cite paragraph vector Mikolov) has marked differences explained after we introduce our model.

To facilitate the prediction of categories a document belongs to, we keep the document embeddings learnt earlier constant and use them as document features to learn category embeddings using the training data.

2.1 Model for learning Document and Word Embeddings

Given a set of documents, D and a vocabulary of words, V built using the words in the given documents, we aim to represent each document and word with a d-dimensional vector representation (or embedding), for example, represented by w_{d_i} for the i-th document in the set of documents. The set of all word and document embeddings is contained in the set W.

Our model, given embeddings of words in a context, tries to predict the middle word in the context using the word embeddings and the embedding of the document, the context belongs to. Formally, given embeddings, (w_{t-c}, ..., w_{t-1}, w_t, w_{t+1}, ..., w_{t+c}), of a sequence of words in a context of size $2c + 1$ words and the embedding of the document, w_{doc}, the context belongs to, we wish to use the embeddings, (w_{t-c}, ..., w_{t-1}, w_{t+1}, ..., w_{t+c}), of the words surrounding the middle word and the document embeddings, w_{doc}, to predict the middle word, i.e., predict the embedding, w_t, of the middle word.

To estimate the probability of the middle word, given the context and the document, we develop a neural network arhitechture in the following manner,

1. The embeddings of the document and the context words are combined to form a hidden layer vector with h-hidden nodes. It is done by, concatenating the document embedding and the context word embeddings in order, in the following manner.

$$\mathbf{x} = (w_{doc}; w_{t-c}; \ldots; w_{t-1}; w_t; w_{t+1}; \ldots; w_{t+c}) \tag{1}$$

where, $\mathbf{x} \in \mathbb{R}^{(2c+1)d}$, is a vector that is constructed by concatenating the document and the context embeddings. The hidden layer vector is constructed by taking a weighted average of the individual elements of the concatenated vector $\mathbf{x}$.

$$h_c = H_c\mathbf{x} \tag{2}$$

where, $H_c \in \mathbb{R}^{h \times (2c+1)d}$ is the matrix of the weights and $h_c \in \mathbb{R}^h$ is the hidden layer constructed from the document and the context word embeddings.

2. Similarly, a hidden layer vector h_t is constructed using the middle word in the context, whose probability of existence is to be estimated. It is done by taking a weighted average of the elements of the middle word embedding,
$$h_t = H_t w_t \tag{3}$$
where, $h_t \in \mathbb{R}^h$ is the hidden-layer vector constructed using the middle word in the context, $H_t \in \mathbb{R}^{h \times d}$ is the matrix of the weights and $w_t \in \mathbb{R}^d$ is the embedding of the middle word.

3. Now, the probability of the middle or the t-th word in the sequence, given the context and the document embeddings is estimated as follows,
$$P\left[t = w_t | \mathbf{x}\right] = g(h_c, h_t) = \sigma(h_t \cdot h_c) \tag{4}$$
where $\sigma(s) = \frac{1}{1+e^{-s}}$ is the usual logistic sigmoid.

Now, to estimate the parameters of the model, i.e. the word and document embeddings and the weights of the neural network, we can maximize the likelihood of observing the training data. Before that, let us look at the calculation of the hidden layers in more detail.

The hidden layer corresponding to the context window given in Eq. 2 can be further simplified by discociating the weights for the different context locations and the document embedding in the following manner,

$$h_c = H_c\mathbf{x} = H_{doc}w_{doc} + H_{w_{t-c}}w_{t-c} + \ldots + H_{w_{t-1}}w_{t-1} + H_{w_{t+1}}w_{t+1} + \ldots + H_{w_{t+c}}w_{t+c} \tag{5}$$

where, H_{doc}, $H_{w_{t-k}} \in \mathbb{R}^{h \times d}$. Now, $h_t \cdot h_c$ can be written as,

$$h_t \cdot h_c = w_t^T H_t^T (H_{doc}w_{doc} + H_{w_{t-c}}w_{t-c} + \ldots + H_{w_{t-1}}w_{t-1} + H_{w_{t+1}}w_{t+1} + \ldots + H_{w_{t+c}}w_{t+c}) \tag{6}$$

$$h_t \cdot h_c = w_t^T (H'_{doc}w_{doc} + H'_{w_{t-c}}w_{t-c} + \ldots + H'_{w_{t-1}}w_{t-1} + H'_{w_{t+1}}w_{t+1} + \ldots + H'_{w_{t+c}}w_{t+c}) \tag{7}$$

where, H'_{doc}, $H'_{w_{t-k}} \in \mathbb{R}^{d \times d}$ are the new consolidated weights of the neural network. Hence, now the parameters of our model are the document and word embeddings, W and the new weights given in Eq. 7. The set of the parameters is denoted as,

$$\Theta = (W, H'_{doc}, H'_{w_{t-c}}, \ldots, H'_{w_{t-1}}, H'_{w_{t+1}}, \ldots, H'_{w_{t+c}}) \tag{8}$$

From Eq. 4, we see that, the estimate of probobability of the t-th word to be w_t is given by,

$$P_\Theta\left[I_{t=w_t}\right] = \sigma(h_t \cdot h_c)^{I_{t=w_t}} (1 - \sigma(h_t \cdot h_c))^{(1 - I_{t=w_t})} \tag{9}$$

where $I_{t=w_t} = 1$ if the t-th word $= w_t$, else $I_{t=w_t} = 0$.

2.2 Learning the Parameters

The training data, $\mathcal{T}$ is composed of M training instances, each of which consists of a $2c+1$-length sequence of words forming a context and the document it belongs to. For example, $t = \{w_{doc}^{(m)}, w_{t-c}^{(m)}, \ldots, w_t^{(m)}, \ldots, w_{t+c}^{(m)}\}$ represents the m^{th} training instance. For each of the training instances we can estimate the probability of the middle or the t-th word given the context and the document using Eq. 9. To estimate the parameters Θ, we maximize the log likelihood of the training data.

$$\hat{\Theta} = \arg\max_{\Theta} \; l(\mathcal{T}, \Theta) \tag{10}$$

$$l(\mathcal{T}, \Theta) = \sum_{m=1}^{M} \log P_\Theta \left[I_{t^{(m)}=w_t^{(m)}} \right] \tag{11}$$

where,

$$\log P_\Theta \left[I_{t^{(m)}=w_t^{(m)}} \right] = I_{t^{(m)}=w_t^{(m)}} \log \sigma(h_t^{(m)} \cdot h_c^{(m)}) + (1 - I_{t^{(m)}=w_t^{(m)}})(1 - \log \sigma(h_t^{(m)} \cdot h_c^{(m)})) \tag{12}$$

To maximize the log likelihood, we will use Stochastic Gradient Ascent updates as derived below. The derivative of the log probability estimate in Eq. 12 w.r.t. to a paramter $\theta \in \Theta$ is given by,

$$\frac{\partial \log P_\Theta \left[I_{t^{(m)}=w_t^{(m)}} \right]}{\partial \theta} = \left[(I_{t^{(m)}=w_t^{(m)}}) \frac{1}{\sigma(s)} - (1 - I_{t^{(m)}=w_t^{(m)}}) \frac{1}{(1-\sigma(s))} \right] \frac{\partial \sigma(s)}{\partial \theta} \tag{13}$$

$$= \left[(I_{t^{(m)}=w_t^{(m)}}) \frac{1}{\sigma(s)} - (1 - I_{t^{(m)}=w_t^{(m)}}) \frac{1}{(1-\sigma(s))} \right] [\sigma(s)(1-\sigma(s))] \frac{\partial s}{\partial \theta} \tag{14}$$

$$= \left[(I_{t^{(m)}=w_t^{(m)}}) - \sigma(s) \right] \frac{\partial s}{\partial \theta} \tag{15}$$

where $s = h_t^{(m)} \cdot h_c^{(m)}$ is given in Eq. 7. Therefore

$$\frac{\partial \log P_\Theta \left[I_{t^{(m)}=w_t^{(m)}} \right]}{\partial \theta} = \left[(I_{t^{(m)}=w_t^{(m)}}) - \sigma(h_t^{(m)} \cdot h_c^{(m)}) \right] \frac{\partial (h_t^{(m)} \cdot h_c^{(m)})}{\partial \theta} \tag{16}$$

Now,

$$\frac{\partial (h_t^{(m)} \cdot h_c^{(m)})}{\partial w_t^{(m)}} = \left[H'_{doc} w_{doc}^{(m)} + H'_{w_{t-c}} w_{t-c}^{(m)} \ldots + H'_{w_{t+c}} w_{t+c}^{(m)} \right] \tag{17}$$

$$\frac{\partial (h_t^{(m)} \cdot h_c^{(m)})}{\partial w_{t-k}^{(m)}} = H'_{w_{t-k}} w_t^{(m)} \tag{18}$$

$$\frac{\partial (h_t^{(m)} \cdot h_c^{(m)})}{\partial H'_k} = w_t^{(m)} (w_k^{(m)})^T \tag{19}$$

The update to be made to a parameter $\theta \in \Theta$ on observing a training instance m on the $i+1$-th iteration is,

$$\theta^{(i+1)} \leftarrow \theta^{(i)} + \gamma \left[\frac{\partial \log P_\Theta \left[I_{t^{(m)} = w_t^{(m)}} \right]}{\partial \theta^{(i)}} \right] \tag{20}$$

where gradients of different kinds of θ are given in Eq. 17, 18 and 19 representing gradient of the middle word embedding, gradient of the embedding of a word in context or document embedding and gradient of the weights repectively.

2.3 Differences with Paragraph Vector Model of Mikolov

The paragraph vector model (PV-DM) (cite mikolov para) is very similar to our model, though it contains several differences in the way their model is formulated.

The PV-DM also maximizies the average log probability of predicting the middle word given the sequence and the document it belongs to. They estimate the probability via a multiclass classifier, such as softmax.

$$p(w_t | w_{para}, w_{t-k}, \ldots, w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_{i=1}^{|V|} e^{y_i}} \tag{21}$$

Where each of y_i is un-normalized log probability of each output word i, computed as

$$y = b + Uh(w_{para}, w_{t-k}, \ldots, w_{t+k}; W) \tag{22}$$

where U, b are the softmax parameters and h is constructed by average of context word vectors.

The differences between our model and PV-DM is that, the hidden layer in PV-DM is constructed by averaging the context word vectors because of which it loses syntactic information and reduces to a model similar to a bag-of-words. Our model on the other hand consists of separate weight matrices for each of the context locations while constructing the hidden layer to encode the syntactic information in the hidden layer which we expect to help in learning more complex and representative document and word vectors. Also, the weight matrix U in PV-DM contains rows equal to the number of words in the vocabulary and needs to re-sized and re-trained(? is re-training necessary ?) every time a new word is added to the vocabulary.

3 Experimentation

Our experimentation involves learning document embeddings using the technique we described and then using them to learn category embeddings to perform multi-class document-classification.

Apart from held-out evaluations we would also present cold-start evaluation where no information about the categories of the documents in the test set is available and we evaluate the performance of our method on categorizing such documents.

The other, even more challenging task is to learn the embeddings of the new documents only given fixed embeddings of the words and no other documents.

3.1 Baselines

There are a bunch of baselines that we will compare our method to.

1. **Trivial Matrix Factorization** : Here we trivially factorize the document-category matrix using the probabilistic matrix factorization formulation shown above. We expect this method to perform well on the held-out evaluation task but it faces challenge of learning document embeddings in the cold-start evaluation.

2. **Effect of using Doc Embeddings as Pretrained Vectors** : Use the doc embeddings learnt from our method as doc embeddings initialization for matrix factorization

3. **Bag-of-words** : In this method we would represent our documents with sparse feature vectors that are the size of the vocabulary and the features can only take two possible values, 0 or 1 representing the absence or presence of the particular word in the document. The values 0/1 can also be substituted by tf-idf values.

4. **Word-Vector Average** : In this method, during learning we would only use the context word embeddings to predict the middle word and hence learn word embeddings. Using these embeddings we will represent the document as the average of the word vectors for the words that are present in the document.

3.2 Models for Document Embeddings Learning

Compare various variations of our model on their efficiency of categorizing documents coorectly and also compare the space and time complexity requirements of the different models. The different models being,

1. **Averaging Context Words** : In this model we would only take an average of the word vectors and the document embedding to find the embedding of the hidden layer corresponding to the context. This corresponds to the weight matrices, H in our model to be identity matrices that would not be updated during training.

2. **Weighted Average Context Words** : In this model, to find the hidden vector for the context we take a weighted average of the context word and document vectors. This corresponds to the weight matrices, H in our model to be, diagonal matrices whose diagnols entries would only be updated and remain the same always. This is also equivalent to having a scalar weight instead of matrices.

3. **Weighted Transformation Average** : In this model, the weight matrices, H would be initialized to some random quantity and will be updated during the learning phase. One variation to this model is to initialize the weight matrices to be identity and then update them during learning.

3.3 Hyperparameter Tuning

Our method, for both, learning document embeddings and multi-label document classification contain many hyperparameters that need to be tuned for optimum performance. The hyperparameters that need to be tuned for each model are,

- **Learning Document Embeddings** :

  1. **Epoch** : The number of passes that the document embeddings learning algorithm makes over the training data is an important parameter that needs to be tuned for precise document and word embeddings to be learnt.
  2. **Embedding Size** : The number of dimensions that the documents and words need to be embedded to, is a crucial parameter, since embedding into a lower dimension space may lead to loss of informationa and embedding in a high dimensional space mein introduce redundancy.
  3. **Negative Samples** : Our method's training relies on negative sampling corrupt center words in contexts, the number of which per correct context needs to be tuned.
  4. **Window-size** : The optimum window size of the context that is considered needs to be empirically found.
  5. **Sub-sampling frequent words** : One intuition to learn efficient and more meaningful embeddings is to not consider words in contexts that occur very frequently. This usually includes stop-words or words that occur frequently in a domain. Such words are considered to not contribute to the semantic content of the document and hence can be neglected.
  6. **Learning Rate and Regularization Constant** : The learning rate and regularization constant used needs to be tuned.