EECS 595 - Semi-Supervised learning of Patterns in Text using LSTM

Cyrus Nikolaidis, Hrishikesh Rao, Aravind Bharathy, and **Graham Palmer**

Abstract

Classifying text samples is a classic problem in Natural Language Processing, and there is a wide variety of available methods to perform such a task given a labeled dataset. The goal of this project is to attempt to apply these methods in a semi-supervised way - that is, with two sets of samples, one labeled, one unlabeled, as training data, in such a way that the resulting model has better performance than if it were trained on just labeled data. In doing this, we use LSTM, a powerful model for language modeling.

1 Introduction

Several of the topics studied in this course relate to creating models of language - which models a "language" as a probability distribution of possible strings.

We can create these models of language using labeled samples of text, and thus determine the "language" (or, more generally, classification) of an unlabeled sample of text with high accuracy. An example is using a unigram model to guess whether a piece of text is a sentence in English or French, or to predict the sentiment of a piece of text.

An issue with this is that many of these labelings are tedious to create. The goal of this project is to create a classifier that, using a labeled dataset and a unlabeled dataset, can create a classifier of higher quality than what we could create given only the labeled dataset.

This initially seems impossible - what meaningful information do the text samples carry without labels? However, our intuition that this is possible to do is motivated by observing other unsupervised training methods, such as k-means. In this method, cluster centers for a set of points are found through repeated iteration of 1) assigning each point a cluster center, and 2) choosing new cluster centers based on the points belonging to each cluster.

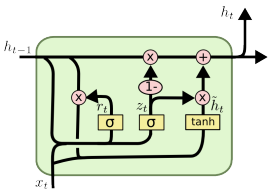
The reason this works is because points in the same cluster have similar features, and the model can recognize this even when the points are not labeled. One can imagine that one can use a language model, which outputs a probability ("a distance") for each class label can be imagined as a cluster center for a set of text samples, and we can repeatedly retrain the language models and label unlabeled text data to achieve a similar result as with the k-means algorithm (indeed, both of these methods would function as a special case of the E-M unsupervised learning algorithm). We could improve this by using a semi-supervised method to initialize the language model using correct data to set it on the "right path" and then use the unsupervised methods to leverage the rest of the data.

As it turns out, there are many ways of applying unsupervised learning methods to text classifications. We will explore an option we find effective in this project.

2 Related Work

There has been extensive work in both the areas of semi-supervised training language models and the use of LSTMs for text classification, although it is not immediately obvious that anyone had published anything about using these two methods in conjunction, as we intend to do here. Our application of both topics here draws heavily from past work on the topics.

(Sundermeyer et al., 2012) is an example of a recent work which explores the use of LSTM neural network in language modeling. Since the time of it's publication, LSTM has become a commonly used model for the original supervised version of this task, so our use of it here is not at all unprecedented, and the exact model architecture can be drawn from



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 1: An example of an LSTM cell, the recurrent component of the model used in this project

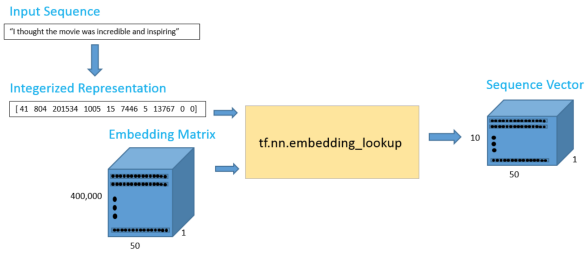


Figure 2: Method of generating sequences using word embeddings.

these past uses.

(Nigam and Kamal, 2000) and (Mihalcea, 2004) explore different methods (E-M algorithm, self-training, and co-training) for solving language problems in a semi-supervised way. The way we are currently planning to execute the semi-supervised learning process is via self-training, described in (Mihalcea, 2004).

3 Methods

3.1 LSTM Model Description

LSTM is a highly effective recurrent neural network structure, and is particularly useful in natural language contexts. (Sundermeyer et al., 2012) In this project we will treat LSTM largely as a black box for a general "language model". That is, when trained on samples of text labeled as being in one of n languages, we assume LSTM outputs for an input string n values proportional to the probability that the string belongs to each of the languages. This can be thought of as simply a classifier.

The method we use to generate sequence predictions (as a set of class label probabilites) with LSTM as follows:

- In order to translate our word sequence data effectively into meaningful states, we use pre-trained word embeddings. These word embeddings are created in an unsupervised manner to contain as much

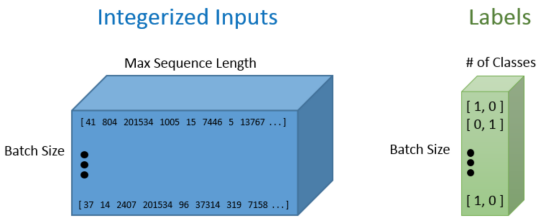


Figure 3: Input and output of LSTM model.

- We pass these sequences through an LSTM to retrieve an output state.
- We compute a set of class probabilities from this state by passing the output state through a linear and softmax layers in succession.

3.2 Semi-Supervised Learning Methods

The algorithm we are planning to use for semi-supervised learning is called self learning, and works as follows:

Data: input sequences, input labels, unlabeled sequences
Result: Model for predicting class labels
threshold := $\in [0, 1]$;
training sequences := input sequences;
training labels := input labels;
i := 0;
while *i < some number of iterations* **do**
  Train LSTM model on training sequences, training labels;
  Predict label and confidence score for each sequence in unlabeled sequences;
  for *each unlabeled sequence* **do**
    if *confidence > threshold* **then**
      Add sequence, predicted label to training sequences;
    end
  end
  i += 1;
end

Algorithm 1: Self Learning

In other words this algorithm repeatedly trains a classifier, predicts labels for unlabeled data, and then adds the predicted labels for data for which it is most confident about to the training data.

4 Datasets

4.1 Text classification datasets

The goal is to find datasets containing samples of text annotated with some small finite set of class labels. Due to the limitations of our model, it is necessary to find datasets which are sufficiently large (at least 10000 examples) and for which the differences between sentences are not so subtle that the supervised LSTM model cannot learn to distinguish between classes.

The dataset we are currently leveraging is the **IMDB Movie Reviews dataset**, a dataset of 25000 examples of movie reviews labeled by sentiment (half positive and half negative). Given that users submit ratings along with the review texts, manual data annotation was not necessary here. We have observed that this works well with our original supervised LSTM method (81% test set accuracy) so should be fine to use for the semi-supervised process.

An example of a "positive sentiment" review in the dataset that the model learns to recognize:

```
This is a great movie.
It is based on a true story.
This movie helps not only children
cope with losses,
but older people as well.
Hope everyone will enjoy it!!!
```

In addition to this dataset we are looking into experimenting with a number of other datasets (Newsgroup classifications, Music Genre classifications, etc.)

4.2 Word embeddings

We also obtain our word embeddings from an outside source. Specifically we use pretrained word embeddings by GloVe. (Pennington et al., 2014) of dimension 50.

5 Project Evaluation

For any given dataset, by holding out a portion of the data to be a test set, our models can be compared simply with the metric of classification accuracy, the proportion of the test set for which the model predicts the correct label. (It is important to note here that the test set is disjoint from both the labeled and unlabeled data used to train any given model).

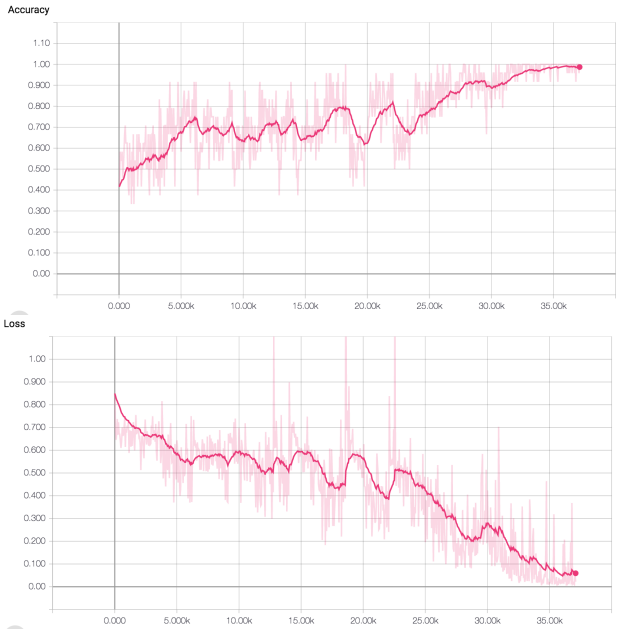


Figure 4: Accuracy and loss of our model when trained on supervised labels on the IMDB movie reviews dataset as a function of training iterations

The most important metrics of success for the project, given this evaluation metric is then:

- Using the Semi-Supervised Learning algorithm, what test accuracy do we obtain on the dataset relative to the same algorithm trained only on the "labeled" data?

- Using the Semi-Supervised Learning algorithm, what test accuracy do we obtain on the dataset relative to the same algorithm trained on both the labeled and unlabeled data in a supervised manner (both using the correct ground truth labels)?

We expect the accuracy of our semi-supervised method to be above the former and below the latter, with the project being more successful if we can get an accuracy closer to the accuracy of the model on the full dataset (that means we can leverage unlabeled data to the same level as labeled data, which seems unlikely)

References

Rada Mihalcea. 2004. Co-training and self-training for word sense disambiguation. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*.

Nigam and Kamal. 2000. Text classification from labeled and unlabeled documents using em. *Machine Learning.*

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Sundermeyer, Schlter, and Ney. 2012. Lstm neural networks for language modeling. *Thirteenth Annual Conference of the International Speech Communication Association.*