Agost Biro
June 5, 2017

Capstone Proposal

Word2Vec for Sentiment Analysis

Domain background

Sentiment analysis is one of the fundamental tasks of natural language processing. It is commonly applied to process customer reviews, implement recommender systems and to understand sentiments in social media. SemEval (short for semantic evaluation) is an annual competition designed to facilitate and track the improvements of semantic analysis systems. The best submissions to the SemEval 2016 Sentiment analysis in Twitter task overwhelmingly relied on word embeddings, the most popular of which was Word2vec. (Nakov et al., 2016)

Vector embeddings are learned vector representations of information such that the constructed vector space exhibits regularities useful for a specific task. Word embeddings are a natural language processing technique where a vector embedding is created from words or phrases. The purpose of these embeddings is to provide a representation of the input that is convenient to operate upon for other natural language processing or information retrieval methods. (Wikipedia: Word embedding) The Word2vec model is popular because it offers state-of-the-art performance in evaluating semantic and syntactic similarities between words with relatively low training times. (Mikolov et al., 2013) I find the topic of vector embeddings interesting, because I suspect that it might be possible to discover new information about a corpus based on the analysis of its embeddings.

Problem Statement

I intend to solve the Bag of Words Meets Bags of Popcorn Kaggle competition as my capstone project. The task involves the binary classification of movie reviews from IMDB as expressing either positive or negative sentiment. The solutions are evaluated by Kaggle.

Datasets and Inputs

The dataset is provided courtesy of Kaggle and it was collected in association with (Mass et al, 2011.):

> The labeled data set consists of 50,000 IMDB movie reviews, specially selected for sentiment analysis. The sentiment of reviews is binary, meaning the IMDB rating < 5 results in a sentiment score of 0, and rating >=7 have a sentiment score of 1. No individual movie has more than 30 reviews. The 25,000 review labeled training set does not include any of the same movies as the 25,000

> review test set. In addition, there are another 50,000 IMDB reviews provided without any rating labels. (Chapman, 2014)

I am going to use the labeled training data set for training and then use Kaggle's service to evaluate my predictions on the testing data. I am going to use the unlabelled training data to produce word embeddings.

Solution Statement

The solution to this problem is to provide a binary sentiment classification (0 for negative, 1 for positive) for each item in the testing data using feature vectors from a self-trained Word2vec model, upload it to Kaggle and then have it scored. The best solution achieves the highest Kaggle-score. I intend to try a number of different approaches as discussed in the Project Design section below.

Benchmark Model

I intend to use feature vectors created with the bag-of-words method using the 5000 most frequent terms and then train a random forest classifier using these feature vectors to predict the sentiment of a review as suggested by Kaggle. In addition to the random forest classifier, I would see how linear regression, logistic regression and a support vector classifier perform on this task.

Evaluation Metrics

Kaggle uses the area under the receiver operating characteristic (ROC) curve to evaluate submissions. An ROC curve is produced by plotting the true positive rate (TPR) against the false positive rate (FPR) at various discrimination thresholds [0, 1] and connecting the points. TPR = TP / (TP + FN) and FPR = FP / (FP + TN), where TP is the number of true positives, FN is the number of false negatives, FP is the number of false positives and TN is the number of true negatives. (Wikipedia: Receiver operating characteristic)

Using the area under the ROC curve as an evaluation metric in this case is somewhat confusing, as the submission should only contain a binary classification (ie. 0 or 1) and not the probability of a review belonging to one class or the other. So in this case, only one point will be plotted after calculating the TPR and FPR on the submission with an unknown threshold value from the evaluator's point of view. The curve is then completed by connecting the points (0, 0), (TPR, FPR) and (1, 1). (Source: Kaggle forums, Ilya Ezepov and franck)

Project Design

I intend to follow the general project design outlined by Kaggle in their tutorial with additions of my own.

Cleaning up the reviews and validation set

The training data is found in tab-separated CSV files. It can be parsed with the Pandas library. The reviews in the dataset contain HTML markup and have to be cleaned up with the Beautiful Soup library. In addition, punctuation has to be removed and a list of lowercase words has to be produced from each review with standard string manipulation tools. The last step is to remove stop words from the review with the NLTK library. In addition to what's suggested by Kaggle here, I intend create a validation set using 20% of the training data to evaluate the results of parameter tuning.

Creating the baseline models

The first step in creating the baseline models is producing the feature vectors using the bag of words method with the help of the scikit-learn library's CountVectorizer. Then, a binary classification model can be trained on the thus produced feature vectors to classify the user reviews. Kaggle suggests using a random forest classifier (RFC) here. In addition to that, I am going to try out a linear regression model (LR), a logistic regression model (LogR) and a support vector classifier (SVC) as well using a radial basis kernel function, as it would be interesting to see how different models with different assumptions about the data perform. (LR assumes the data is separable with a single line, LogR assumes there is a single decision boundary that can take on complex shapes, RFC assumes there are multiple decision boundaries that are axis aligned and SVC assumes that there are multiple decision boundaries that can take on complex shapes.)

Word2vec models

I am going to use the unlabeled reviews to produce the word embeddings, as suggested by Kaggle. The reviews should be cleaned up as before, but stop words shouldn't be removed. Deviating from the Kaggle tutorial, I intend to use the TensorFlow library to implement the neural network that produces the word embeddings with the hierarchical softmax training algorithm, as it has a better semantic accuracy score than the continuous bag of words model according to (Mikolov et al, 2013) using a vector dimensionality of 300. The other training parameters are to be determined.

In addition to my own trained model, I am also going to try out the Word2vec model published by Google that contains 300-dimensional vectors for 3 million words and phrases.

Classification using Word2vec models

The challenge of using a Word2vec model for the classification of the reviews is getting from word vectors to feature vectors for the reviews. Here, I intend to follow the suggestions of the Kaggle tutorial first and try creating the feature vectors at first by simply averaging them, then introducing weighing with by the tf-idf statistic. The next suggested technique is clustering the words with K-means and creating “bag of clusters” feature vectors by counting the frequency of the words that belong to a cluster in a review. The rationale behind this would be reducing the dimensionality of the feature vectors. Besides K-means suggested by Kaggle, I also intend to try out affinity propagation which is smart enough to figure out the number of required clusters on its own. An additional technique that would interesting to try out to produce the feature vectors is scaling the word vectors in a review by their tf-idf weight and then summing them up.

Then, using the feature vectors produced for the reviews, I am going to train and test the four models binary classification models mentioned above (LR, LogR, RF, SVC). I also intend to try different different discrimination thresholds on the validation data.

References:

Chapman, Angela. *Bag of Words Meets Bags of Popcorn | Kaggle.* N.p., 2014. Web. 03 June 2017.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). "Learning Word Vectors for Sentiment Analysis." *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).* (link)

Nakov, Preslav, et al. "SemEval-2016 task 4: Sentiment analysis in Twitter." *Proceedings of SemEval* (2016): 1-18.

Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

Wikipedia contributors. "Receiver operating characteristic." *Wikipedia, The Free Encyclopedia.* Wikipedia, The Free Encyclopedia, 25 May. 2017. Web. 1 Jun. 2017.

Wikipedia contributors. "Word embedding." *Wikipedia, The Free Encyclopedia.* Wikipedia, The Free Encyclopedia, 24 Apr. 2017. Web. 27 May. 2017.