The Refugees: Bag of Words Meets Bags of Popcorn

Problem Description

The competition we chose is named Bag of Words Meets Bags of Popcorn and is accessible on Kaggle. It is intended as a tutorial competition to get started with natural language processing. The main goal is to predict the sentiment of movie reviews taken from IMDB. The sentiment is binary and submissions are judged on area under the ROC curve. We were given three data sets. The two labeled data sets each consist of 25,000 IMDB movie reviews, specially selected for sentiment analysis. One of them serves as a training set, whereas the other is used as a test set. The third data set consists of another 50,000 IMDB reviews without any rating labels.

Approaches

Data preprocessing

The data we are given is in the tsv file format, containing the id of the review, the sentiment (if it's labeled data), and the raw text of the review. Since the data comes from IMDB, some reviews contain HTML tags. We cleaned the tags using the Beautiful Soup library in Python. We also removed any punctuation and numbers. We did this by matching them to regular expressions using the re library in Python. Finally, we removed frequently occurring words without much meaning, called stop words, using the NLTK library in Python. The average length of the reviews in the training set was 823 words before cleaning, and 119 after.

Distributed Word Vectors

We built distributed word vectors, proposed by Mikolov et al. (2013). We used the word2vec tool to built the vectors, and use the training set, as well as the additional unlabeled dataset, since this method employs an unsupervised learning approach. We used vectors of the size 300, context size 10, the skip-gram model, hierarchical softmax as the training algorithm, and used 40 for the minimum word count. We used the word vectors as features for our classifier in two ways. The first one is to average the word vectors from each review, and end up with a single averaged 300-dimensional word vector per review, and use them as features for a Random Forest classifier. The second one is to group similar vectors together. Our intuition is to first find the centers of the word clusters, called centroids, and then convert reviews into bags-of-centroids. We used the K-Means algorithm to do this, where K is the number of clusters, and we set K = 5. Again, we use these bags-of-centroids as features for a Random Forest classifier. The results of both approaches are reported later.

Features

Additionally, we experiment with different types of features and perform feature concatenation. We create bag-of-words features, TF-IDF features, n-gram features (unigrams, bigrams and trigrams), as well as use a list of positive and negative words, found here, count these in our reviews, and use them as features. We also tried using a regex approach, in which for instance the *nogram* construction, negation+adjective (e.g not great),

was counted and then in combination with the positivity word list assigned to be a positive or negative expression. However, we abandoned this because defined expressions occurred too infrequent, about once in every third review, to have any significant effect. Going to higher-level constructions on the other side, like adjectives against verbs or only the total number of words has been shown to be randomly distributed.

Models

We use several different classifiers: Multinomial Naive Bayes, Bernoulli Naive Bayes, Random Forest Classifier, and a semi-supervised ensemble of classifiers. In the semi-supervised ensemble, the second model uses the output of the first model as its input. We split the training set into two parts, ratio 70:30, and use the larger one for model training, and the smaller one for model evaluation.

Results

Model	Features	Score on Train	Score on Test	Score on Kaggle
Majority Class Classifier	-	.5	.5	.5
Random Forest	BoW	1	.84	.844
Random Forest	Averaged vectors	1	.837	.833
Random Forest	BoC	1	.506	.842
Semi-supervised MNB	n-gram	.999	.855	.844
Semi-supervised MNB	TF-IDF + n-gram + POS-NEG	.997	.864	.854
BNB	n-gram	.999	.811	.756
BNB	TF-IDF + n-gram + POS-NEG	.998	.838	.793
MNB	n-gram	.999	.881	.854
MNB	**n-gram + POS-NEG**	**.998**	**.882**	**.863**

Discussion

We use the Majority Class Classifier as our baseline. We find that the simplest feature representation, BoW, obtains a high score. It is interesting to see that even word vectors do not outperform BoW on this task. Also, it seems that our ensemble approach did not boost performance. However, we observe that introducing a custom-made feature (POS-NEG), in combination with n-grams, yields the best result. A score of .863 places us in the 289th place, out of 578.

Contributions

Our team consists of five team members. The contributions of individual team members are listed below, sorted alphabetically.

Csaba Balint:

- Improved Paula's original solution to
  - Include cache-ing of cleaned reviews (it took 10 or more minutes of runtime)
  - Optimized for less memory usage, as it was a constant problem (eg. using sparce matrix)
  - Modularized the code to make it easy to implement new feature detectors and fitting algorithms
  - Creating a final submission is as easy as changing train-test ratio from 0.7 value to 1.0
- Created a feature that concatenates others
  - Any number of features can easily be concatenated into a single one
- Implemented Semi-supervised learning class
  - It takes another learning model or two (the second is by default equal to the first)
  - The second model is also trained on the output of the first on the unlabelled data

Adam Barillas:

-

Dennis Doerrich:

- Created and tested regular expression based *nogram* feature based on the spaCy natural language processing package
- Implemented n-gram feature

Paula Gombar:

- Set up pipeline and wrote basic modules (preprocessing, feature construction, model selection)
- Implemented BoW and TF-IDF with MNB and BNB
- Implemented word vector representations

Anna Tokes:

- Created a feature which counts the positive and negative words in the reviews
- Implemented a feature based on regular expressions

The source code can be found on GitHub.

References

1. 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).
2. Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

3. Maas, Andrew L., et al. "Learning word vectors for sentiment analysis." Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011.