Abstract

In this paper, we explore how to predict the emotional response distribution of an article with text and images, based solely on the article’s text. In particular, we establish a new data corpus of BuzzFeed articles. We run Bayesian estimators with dependency parses and neural networks on this dataset and find that while dependency parses with linear models (Jensen-Shannon (J-S) distance 0.319, lower is better) do not improve over our baseline prediction (J-S distance 0.286), the recurrent neural network (RNN) is able to significantly outperform the baseline prediction (J-S distance 0.248). Since this is a novel dataset, more testing with this corpus is necessary to determine if our results are competitive.

1 Introduction

Predicting and understanding sentiment in response to content is valuable for deeper understanding of reviews, social networks, blogs and news websites. Sentiment analysis has proven a valuable source of progress in natural language processing and natural language understanding. While there has been a wealth of research into predicting positive and negative sentiment, the picture is in reality more nuanced and many emotions can come up in response to a particular piece of text. We wish to explore the full emotional reaction distribution in reaction to text articles.

Prior literature on the subject of sentiment analysis has found that although relatively simple approaches such as Naïve Bayes and support vector machines (SVM) with unigram and bigram (bag of words/bigrams, or BOW) features are reasonably accurate for predicting binary sentiment, BOW features are typically inadequate for more complex sentence constructions and difficult to improve upon (Pang et al., 2002). Accordingly, such superficial features will not suffice for our task: we must apply more sophisticated natural language understanding (NLU) techniques to successfully predict a more nuanced distribution. Similarly, simply using semantic orientation to predict binary sentiment performs comparably to the other BOW features, with the same issues in generalizing to more nuanced analyses of emotion (Turney, 2002). Neural networks are far more powerful, can encode more linguistic information, and typically perform better, but they come at the cost of added complexity. There are many different methods and architectures for creating neural nets, each of which has different characteristics and performance. Regardless of the approach, distinguishing between more than two classes (i.e., binary sentiment) is difficult, and accuracy drops significantly when doing so.

In this paper, we introduce a new dataset of BuzzFeed articles, annotated by readers with the emotional reaction distribution, and use this dataset to train and test our models. With this dataset, we explore a combination of supervised (Maximum Likelihood Estimate (MLE) and Maximum A Posteriori (MAP) estimators with dependency parses) and semi-supervised learning approaches (neural networks and unsupervised word vectors).

2 Dataset

We scrape BuzzFeed for data on 16,426 articles, which represents all the articles that we can retrieve from the website’s “Buzz” and “Community” sections. For each article, users can vote for up to three out of the thirteen categories: “cute,” “drab,” “ew,” “fab,” “fail,” “hate,” “lol,” “love,” “omg,” “trashy,” “win,” “wtf,” and “yaaass.” While BuzzFeed content is centered around images, it is of interest to see how accurately we can predict the
correct reactions based purely on the text in the article.

After filtering BuzzFeed articles for those that received at least 50 votes, we are left with 12,685 articles. Of these, around 75% of these articles’ most popular emotion is “love,” and the most popular emotion for nearly every other article is “lol.” We notice that the scraped pages also contain British versions of the reaction categories, which are partially different (they have “amazing,” “blimey,” “splendid,” and “ohdear,” while they exclude a few of the American labels), but the British labels have far fewer votes than the American labels; accordingly, we ignore these categories. Figure 1 shows an example of a web scraped article.

```json
{
  'title': 'The 23 Most Painfully Awkward Things That Happened In 2014',
  'section': 'Culture',
  'body': 'This guy mastering the art of the backup: Riff Raff’s darkest moment:...'
}
```

Figure 1: An example of a web scraped BuzzFeed article. Note that no image data is gathered here, just text.

Importantly, the rated emotions have relationships between each other in terms of reactions, with some emotions (like “ew” and “trashy”) having positive correlations and others (like “lol” and “love”) having negative correlations. This relationship among common BuzzFeed emotions is shown in Figure 2.

3 Baseline Predictor Performance

3.1 Classification

We initially train classifiers to predict the top voted emotion for a particular article in order to explore the topology of our dataset. The two most popular top emotions are “love” and “lol.” In fact, we find these two emotions are relatively orthogonal: they have a strong negatively correlation. Accordingly, we try a variety of classifiers to distinguish between them. There are 1,880 articles with “lol” as the top emotion, and we select a random subset of 1,880 articles with “love” as the top emotion to achieve a 50% split. We use a 66%-34% train-test split.

We use the scikit-learn (Pedregosa et al., 2011) implementation of the following algorithms, and unless otherwise noted default parameters are used.

3.1.1 Features: TF-IDF

The features used for all of the classifiers are a Term Frequency–Inverse Document Frequency (TF-IDF) reweighting of the token counts of the body of each article. TF-IDF computes the weight of each word as the count (term frequency) scaled by the log of one over the number of times that word appears in the corpus (inverse document frequency). Intuitively, we punish words that occur too frequently (such as “the”), since they are likely to carry less intrinsic meaning.

3.1.2 Gaussian Naïve Bayes

Naïve Bayes builds a Bayesian probability model of the features, assuming they are independent when conditioned on the class (i.e., top emotion) of each article. This achieves a score of 0.59 on our dataset. Although this does better than random guessing, it is not much better, suggesting that the Naïve Bayes assumption may not hold well for our dataset. The training accuracy is 0.98, strongly indicating that this model is overfitting.
3.1.3 Logistic Regression

Logistic Regression models the probability of each of the classes as the sum of the features using a logistic function. This has a training score of 0.90 with a testing score of 0.65. Although it is still overfitting, it is not as egregious as with Gaussian Naïve Bayes. More data would likely help so that the model has better information on which features generalize to many articles and which do not.

3.1.4 Support Vector Classifier (SVC)

Support Vector Machine classification works by finding points that can be used as "support vectors" to define the classification boundary between different classes. We found a linear kernel to work best. This has a training accuracy of 0.94, and a testing accuracy of 0.67. Again, this is overfitting (although using a linear rather than Gaussian kernel reduces the amount of overfitting), and so more data would likely help, but this is the best model that we found.

We could of course have spent more time tuning this to get better numbers, but we chose not to since this is just our baseline and our primary goal is to predict the full distribution. That said, these scores are relatively low, indicating that this is a difficult task. Although the models clearly do better than random, and thus are able to learn something from the data, they do not perform well overall. We suspect this is partially because BuzzFeed focuses primarily on images and GIFs, with relatively little text for these models to learn from. To test this, we try a human oracle.

3.1.5 Human Oracle

Using all of the extracted information in a BuzzFeed article, a human subject predicts whether a particular article has “love” or “lol” as its top emotion to see how well actual natural language understanding and higher level reasoning performs on our dataset. Out of 100 articles, he achieves a score of 0.73. This is better than the best machine learning model score of 0.67, but is not significantly higher. While some articles are quite clearly destined to have “love” or “lol” as their top emotion based on the title alone (e.g., 13 Reasons Why “Caitlin’s Way” Was Actually The Best Show From Your Childhood is clearly “love,” as are most posts talking about TV shows, while This Guy Took A Joke With His Friend So Far That He Bought Himself A Billboard is clearly “lol”)), most are rather unclear and the top emotion seems almost arbitrary (e.g., 16 Hilarious One-Star Reviews Of Childrens Books has “hilarious” in its title; nevertheless “love” was the top emotion for this article).

However, trying to predict the full distribution should actually help with this. The aforementioned article with “hilarious” in its title has “love” as its top emotion, but “lol” is quite a close second. As such, if a model predicts the full distribution and guesses that “lol” will be the top emotion, this will not be treated as a large error.

3.2 Predicting the Mean

Our baseline for the full distribution is simply to always predict the average probability distribution. Surprisingly, this performed quite well, achieving a J-S distance of 0.286.

3.2.1 Jensen-Shannon Distance (J-S Distance)

We adopt J-S distance as our evaluation metric, which provides a measurement of the difference between two probability distributions. The J-S distance \( \mathcal{J} \) over two probability distributions \( P \) and \( Q \) is given by the expression

\[
\mathcal{J} = \frac{1}{2} D(P||M) + \frac{1}{2} D(Q||M),
\]

where \( M = \frac{1}{2}(P + Q) \) and \( D(P||Q) \) is the K-L divergence of \( P \) and \( Q \):

\[
D(P||Q) = \sum P(i) \ln \frac{P(i)}{Q(i)}.
\]

We choose J-S distance because it is symmetric, in contrast to the Kullback-Leibler divergence metric.

4 Dependency Parse

A dependency parse aims to capture the directed connections between linguistic units (such as words) in a sentence to extract meaning beyond a simple parse tree. When creating the dependency parse of a sentence, one transforms the string of words into a list of directed word pairs annotated with a relationship. For example, the sentence “It is so beautiful” returns the parse

```
["root", 'ROOT', 'beautiful'],
["nsubj", 'beautiful', 'It'],
["cop", 'beautiful', 'is'],
["advmod", 'beautiful', 'so'],
```

reflecting that ‘beautiful’ is the root of the sentence, ‘It’ is the nominative subject of ‘beautiful’

sentence “It is so beautiful” returns the parse

```
["root", 'ROOT', 'beautiful'],
["nsubj", 'beautiful', 'It'],
["cop", 'beautiful', 'is'],
["advmod", 'beautiful', 'so'],
```

reflecting that ‘beautiful’ is the root of the sentence, ‘It’ is the nominative subject of ‘beautiful’
(since ‘is’ is a copular verb), ‘beautiful’ is the compliment of the copular verb ‘is,’ and ‘so’ is the adverbial modifier of ‘beautiful’ (Manning et al., 2014). Our initial work processes the BuzzFeed article body and produces dependency parses, which we use as features in our predictions.

4.1 Dependency Parse Features

The dependency parses may be considered at several levels in a hierarchical manner. At the root level, we consider the full dependency (for example, the tuple (‘advmod’, ‘very’, ’much’)). However, this feature set is often too sparse, particularly when the data varies significantly. In such cases, dependencies such as (u’appos’, u’delilah’, u’cat’) can appear, which will have a negligible impact on our model due to its rarity. Therefore, we make the dataset more dense by considering more features such as (‘advmod’, ‘very’, 1), (‘advmod’, ‘much’, 2), and (‘very’, ’much’, 3). This level of the hierarchy (the latter two sets) tells us about the function of each word in the dependency and what two words are being compared. The final level of the hierarchy is the densest, yet least descriptive. Namely, only single words (and their positions on the parse) comprise these features. The features demonstrate the tradeoff between feature density (as opposed to sparsity) and descriptiveness (as opposed to ambiguity). In our model, we count how often each of these features shows up in the body of a BuzzFeed article and weight the features by their frequency.

4.2 Dependency Parse MAP (DPMAP)

The dependency parse MAP method works in a similar manner to multinomial Naïve Bayes classification. In particular, in the analysis of dependency features, we consider the Naïve Bayes assumption that different dependency features in the same sentence are independent of each other. Consider the BuzzFeed article x, composed of sparse features representing dependency features from the article. Then the MLE of the multinomial distribution (with Laplace smoothing parameter λ) for the emotion C_k will take the form

\[ p(C_k | x) \propto \sum_{d \in x} p(C_k, d). \] (4)

To train the data, we estimate the distribution \( p(C_k | d) \) by iterating through the articles and tallying votes for emotions and dependencies. We then normalize the probabilities such that \( \sum_k p(C_k | d) = 1 \) (i.e., each dependency has an emotion distribution attributed to it). To test the data, we employ Equations 3 and 4, which yield the emotional distributions for the articles.

To improve the performance of the MLE algorithm, we can add a MAP estimate to bias the performance of the parser toward the a priori distribution derived from \( p(C_k) \) in the training step. This gives the following:

\[ p(C_k | x) = \frac{p(x, C_k) + \lambda}{\sum_k p(x, C_k) + \lambda}. \] (5)

This is important as it now enables us to determine how far predicted distributions fell from the original MAP estimate.

5 Neural Networks

Neural networks are particularly appropriate for this task because of their ability to capture complex interactions and because an output layer with multiple dimensions can be used to predict the full emotion distribution, in a single model. Each output node is used to represent one emotion. When using a prediction from the neural net, any negative predictions are assumed to be 0 and the remaining values are normalized to produce a probability distribution.

5.1 Shallow Neural Network (SNN)

A shallow neural network is one that has a single hidden layer. For the shallow neural net, we used unsupervised GloVe vectors (Pennington et al., 2014), which are continuous fixed-length vector representations of words that have been shown to encode some semantic meaning, as input features; the feature representation of the words in a phrase is the average of the corresponding word vectors. If we use multiple different texts as input to the neural network (e.g., both title and subtitle), the features for all of them are simply concatenated together.

We search a variety of different approaches to the shallow neural net:
• different activation functions
• different combinations of the title, subtitle and body as input features
• different pre-trained GloVe vectors, of different dimensions
• instead of GloVe, using pre-trained word2vec vectors (Mikolov et al., 2013) of 300 dimensions that were trained on 100 billion words from a Google News dataset
• different numbers of hidden dimensions in the hidden layer
• different number of iterations

Most of these variations yield a J-S distance similar to the baseline’s 0.286. The best approach we have is using the tanh activation function, using the title and subtitle to create input features, 50-dimensional GloVe vectors pre-trained on Wikipedia 2014 and Gigaword 5, 30 hidden dimensions, and 300 iterations. This achieves a J-S distance of 0.277.

5.2 Recurrent Neural Network (RNN)

A recurrent neural network connects some the output of some of the nodes to the input of nodes in a previous layer, forming a cycle. This allows it to exhibit some form of “memory” of the input, thus allowing the neural net to process data sequentially. We use the Passage library (IndicoDataSolutions/Passage, ) to train our RNN. Rather than using GloVe vectors, we employ a word embedding layer as part of the neural net so that the embeddings themselves can be modified as part of training. Every five epochs, the RNN performance is evaluated on a development set, and training is stopped after the J-S distance on the development set started to increase. The model with the best development accuracy is used on the test set.

Similarly to above, we explore a variety of different approaches to the recurrent neural net:
• different activation functions
• different thresholds for the minimum number of times a token has to appear to be included
• different combinations of the title, subtitle and body as input features
• different sizes of all but the last output layer
• different regularizers

The RNN generally does better than the shallow neural net. Our best accuracy is with an initial embedding layer of 256 dimensions, a gated recurrent neural net (Cho et al., 2014) of 256 dimensions, followed by a final output layer of 13 dimensions for the 13 emotions, the sigmoid activation function with binary cross entropy cost, using just the title, and using all tokens as input regardless of how many times they appear in the input. This has a J-S distance of 0.248.

5.3 Alternative Ideas

One of the interesting ideas we have is to subtract the average distribution from each data point so that the values we are trying to predict will be more closely clustered around 0, allowing the neural net to make more fine-tuned and accurate adjustments. However, for both the shallow neural net and the recurrent neural net, this seems to debilitate our performance.

6 Discussion of Results

A comparison of all results (baseline, DPMAP, SNN, RNN) are shown in Table 1. Notably, this table illustrates that RNN has the best performance.

6.1 DPMAP Results

Our DPMAP results (specified over a test set spanning 30% of the web scraped articles) demonstrate that there was too much sparsity in the data set for there to be a significant predictive power in the model. As a result, many of the predicted distributions showed little deviation from the average emotions (baseline). We can therefore report only a marginal improvement over baseline. Some example results of DPMAP prediction are shown in Figure 3.

6.2 Shallow Neural Network

The shallow neural net is able to perform better than our baseline, but only minimally so. It likely is not a complex enough model to be able to extract significant meaning from the BuzzFeed data. In addition, this suggests that averaging word vectors (which is what we use as input to our shallow neural net) does not adequately capture the meaning of a sentence or phrase (e.g., it is not able to capture compositionality information). While it is a relatively dense feature, allowing faster neural net training and presumably reducing variability in
accuracy (since fewer weights have to be learned), more and better features are likely to help.

6.3 Recurrent Neural Network

The recurrent neural net is our best model. Although it performs relatively well, during training we notice that the number of iterations has a large effect on accuracy. At first, more iterations increases accuracy on a separate development set, as expected, but after a relatively small number of iterations (for us, typically around 20) training accuracy continues to improve but testing accuracy gets worse, implying the model begins to overfit the data. While we tried some regularization approaches, they did not seem to help. Having more data, however, would be beneficial since then the training data would be more representative of general patterns, and it would give the model more information to learn from.

7 Conclusion

We present a novel dataset of BuzzFeed articles containing text and a corresponding distribution of emotional reactions. Our baseline predictor of just
Figure 5: A comparison of model predictions to average emotion baseline and actual distributions, ordered by actual “love” emotion response. This heat map demonstrates the prevalence of emotions in 2,537 test articles, showing a tendency for predictors to tend toward the average emotion response. While not immediately noticeable, the RNN shows the most variability in prediction and out of all the models, most closely resembles the actual gradient in the love emotion.

Table 1: The table shows that compared to baseline (average emotion), the RNN performs better whereas the DPMAP method performs worse. **(70/30 split)**

<table>
<thead>
<tr>
<th>Method</th>
<th>$J$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline: Average Emotion</td>
<td>0.283</td>
</tr>
<tr>
<td>Baseline: SNN (+ GloVe)</td>
<td>0.277</td>
</tr>
<tr>
<td>Approach: DPMAP**</td>
<td>0.317</td>
</tr>
<tr>
<td>Approach: RNN</td>
<td>0.248</td>
</tr>
</tbody>
</table>

using the average emotional distribution actually works rather well, producing an average J-S distance of 0.286. Our best model, a recurrent neural network with a word embeddings layer and a gated recurrent neural net layer achieves an improved J-S distance of 0.248. We have also evaluated using dependency parses and shallow neural networks, determining that it is difficult to predict much better than the average distribution.

Future work would involve adding more training data and having more time for a detailed hyperparameter search. The disappointing results of the dependency method (DPMAP) suggest that further processing of raw dependency features (e.g., incorporating GloVe and word embeddings into the representation) may be necessary to improve that particular model. Another potential area for improvement would be to incorporate an “emotion manifold” into our work, which probabilistically models how the different reactions interact with each other. A potentially bigger improvement that would step beyond natural language understanding would be to integrate article images into our models to get a better picture of an article’s content.

References


“IndicoDataSolutions/Passage.” *GitHub.* Web. 10 June 2015.


