Subreddit Classification of Reddit Comments

Joshua Beal
joshbeal@stanford.edu
Nathaniel Eidelson
nathanje@stanford.edu

December 11, 2015

Abstract

Text classification has long remained a cornerstone of Natural Language Processing (NLP); countless models have been introduced to capture sentiment, word-order, context, and semantics. Simultaneously, Recurrent Neural Network (RNN) algorithms have proven tremendously successful at a variety of machine learning tasks in the past decade, as datasets have become larger and larger. In this paper, we present a text classification task and show that bag-of-words achieves the best results of the traditional NLP models. We then present our customized implementation of an RNN, using a specialized word2vec semantic embedding model in an attempt to outperform bag-of-words. Through the completion of this project, we demonstrate that bag-of-words is an incredibly simple and effective classifier for some tasks, but that it can be outperformed with a well-trained RNN and enough data.

1 Introduction

Reddit is one of the world's largest online discussion forums with millions of active users. It is a complex and at times overwhelming mass of past and current information, organized primarily by the notion of subreddits. These subreddits are made and curated by users themselves, and often come to have a unique personality of their own. They vary in their number of posts, active users, and most importantly their sense of humor. Within a particular subreddit users can create new posts, respond to other posts via leaving comments, or respond directly to other comments. The result is a set of comments within a given post having to do with the same topic, which is defined categorically by the overall subreddit. The goal of this project is to predict the subreddit of a random comment, and outperform the surprisingly high achieving bag-of-words model.

2 Task Definition

The task definition can be formally described by first defining the set of all comments in subreddit i as s_i. Thus, the entire database from which we are performing comment classification is $DB = s_1 \cup s_2 \cup ... \cup s_i$. We now define a document d_{ij} as the j^{th} comment belonging to subreddit s_i. Note that due to memory restrictions that we will discuss in the next section, we limited the size of our s_i by imposing a maximum size of n. Therefore we have $s_i = \{d_{i1}, d_{i2}, ..., d_{in}\}$. The task is then to select at comment at random from DB, and classify the comment to a subreddit s_i. Alternatively: given some d_{xj}, predict the value of x.

Evaluation was done by considering the overall accuracy of each classifier, and comparing it to that of bag-of-words trained on the same set of data.

3 Infrastructure

We were inspired to perform this machine learning task due to the release of a massive 1.7 billion Reddit comments earlier this year [4]. For our implementation, we decided to use only a subset of this data: 54 million Reddit comments from the month of May. We found that by limiting the subreddits in S to be only the most active, a single month of data still contained plenty of comments to train the RNN. We obtained this dataset from Kaggle [7] in the form of a SQL database. We applied a simple tokenizer and text sanitizer to the comment data for use in the models we developed. In order to speed up our testing, we provisioned a Linux GPU instance on AWS EC2. This also enabled us to leverage memory-intensive models without refactoring our usage of machine learning libraries. We found it useful to turn on the multi-core flags when learning our feature representations.

4 Approach

We began with the understanding that the task of comment classification can be thought of in two distinct parts: the language representation **model** and the classification **algorithm** that learns to make predictions based on the model.

4.1 Baseline

In constructing our baseline, we resorted to a bag-of-words model as it is ideally suited for such a document classification task. Our bag-of-words implementation uses sci-py to construct a sparse feature vector (limited to 5,000 features) using the vocabulary of the training data. It then trains a weight vector using logistic regression, counting the raw occurrences of words in each comment and minimizing training loss. The result is shockingly high-performing, achieving 60.4% on 7 subreddits with 20,000 comments per subreddit (as seen in Table 1). We observed this to be the case because certain keywords are incredibly indicative of a single subreddit: ”album” indicating ”Music,” ”puppy” indicating ”aww,” and so on. Our results show that on average, 60.4% of comments contain such a keywords. *Where our baseline is lacking, however, is in capturing the semantic context of these words.*

Subreddit	**Precision**	**Recall**	**f1-score**	**support**
aww	0.42	0.76	0.54	7405
technology	0.66	0.52	0.58	7457
movies	0.68	0.60	0.64	7395
Bitcoin	0.71	0.61	0.66	7579
gaming	0.54	0.51	0.52	7542
politics	0.74	0.64	0.68	7593
Music	0.69	0.60	0.64	7529
avg / total	0.63	0.60	0.61	52500

Table 1: n = 25000, bag-of-words, logistic regression. Accuracy = **60.42%**

4.2 Oracle

For the oracle, two humans who are very familiar with Reddit (and are the authors of this paper) attempted the same classification task. When presented with 30 random comments from 7 given subreddits, our oracle achieved 73.34% accuracy (as seen in Table 2). When presented 30 random comments from 30 given subreddits, the oracle achieved 50.00% accuracy. These results were in line with what we were expecting: significantly more accurate than the baseline, although not perfect. No matter the classifier, general comments such as ”no way” or ”that really sucks” remained arbitrarily classified.

4.3 RNN and Word2Vec

After determining a baseline accuracy of 60.4% and an oracle accuracy of 73.34%, our goal was to improve upon the baseline bag-of-words model. We began by testing numerous options: tf-idf, Word2Vec and Doc2Vec with Logistic Regression (LR) and Stochastic Gradient Descent (with both logistic loss and hinge loss). We ultimately settled on using an RNN with Word2Vec. Word2Vec is a Google tool for

Subreddit	**Precision**	**Recall**	**f1-score**	**support**
aww	0.75	1.00	0.86	3
technology	0.00	0.00	0.00	0
movies	0.33	0.50	0.40	2
Bitcoin	0.67	1.00	0.80	2
gaming	1.00	0.80	0.89	10
politics	0.71	0.71	0.71	7
Music	0.75	0.75	0.75	4
avg / total	0.74	0.73	0.73	30

Table 2: Oracle (two humans). Accuracy = **73.34%**

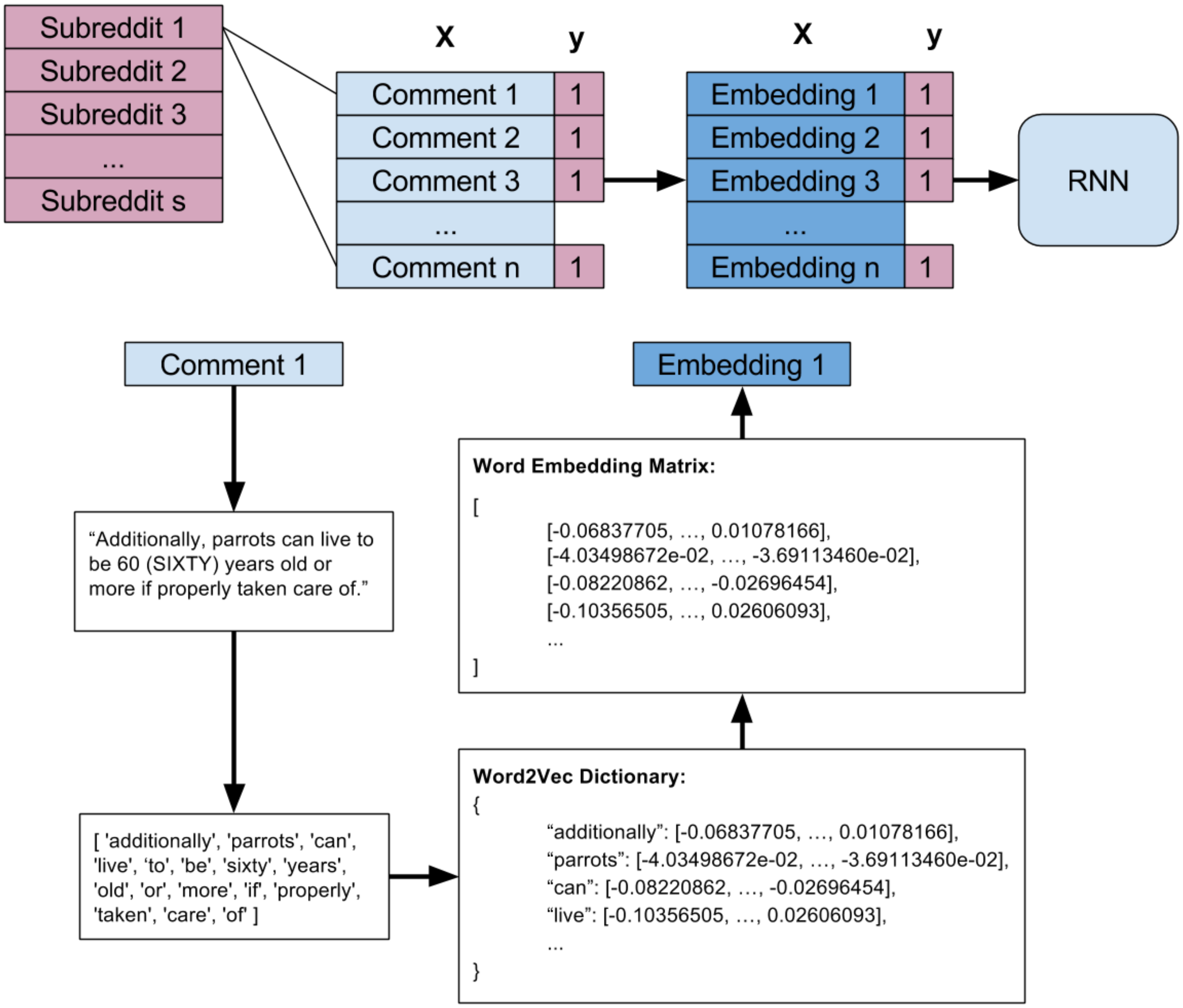
constructing word vectors that goes beyond bag-of-words by capturing the context of each word via a sliding window [6]. This provides an opportunity for an RNN to learn higher-level patterns in comment structure and word order, which we hoped would lead to improved classification over the relatively naive bag-of-words.

The use of Word2Vec enabled us to train our model on a far larger dataset than our RNN, which trained slowly and used more memory. Our best word vectors were trained on 2.6 million comments, and gave us the highest performance with our RNN. In addition, we tried Google's pre-trained word vectors[11] that are trained on 100 billion words but found a slight decrease in performance. This demonstrates what is known: that it is important to train word vectors on the same type of data that they are used to classify. Training our word vectors on 2.6 million comments may be seen as giving the RNN an unfair advantage over bag-of-words. However, we did the same with bag-of-words: training the bag-of-words model with 2.6 million comments while classifying only 25,000 comments gave us 57.15% accuracy, a decrease in accuracy of 3.27%. Thus, we maintain our comparison of our RNN with the baseline bag-of-words model trained on 25,000 comments.

4.4 RNN Structure and Hyperparameters

We aimed to build an effective neural network that could outperform a simple linear classifier. By exploring structures used for other text sequence tasks, we found that recurrent layers often did a good job of encoding the higher-level patterns. We considered both GRU (Gated Recurrent Unit) and LSTM (Long Short-Term Memory) layers for our task, and decided to go with GRU, since brief experiments showed that performance was similar, and GRU often slightly outperforms LSTM on other language tasks [5].

We train the network on a 3D matrix of the comments' word embedding matrices, where each word sequence is capped to length 100 and the null word embeddings are set to 0. The word vectors we used had dimensionality 300. It was necessary to include a masking layer in the neural network and compose the embeddings such that the reset logic is handled correctly (for Keras, this means zero padding on the left, which is counterintuitive from an implementation standpoint). We determined the optimal hidden layer to be of size 200, and the output used the softmax function. We found the "Adam" optimizer to work well for our task with default parameters (little was gained by clipping the gradient norm). We also added a dropout layer, which provides "regularization" for the network and thereby improves overall performance. We found that running for 5 epochs provided good results, and performance started to level off, especially on the validation set, as we ran for more epochs. See "Results & Error Analysis" for our final results after this tuning process. View the figure below to see how these word embeddings are constructed from the raw data and fed as input into the network.



"Additionally, parrots can live to be 60 (SIXTY) years old or more if properly taken care of."

['additionally', 'parrots', 'can', 'live', 'to', 'be', 'sixty', 'years', 'old', 'or', 'more', 'if', 'properly', 'taken', 'care', 'of']

```
Word2Vec Dictionary:
{
        "additionally": [-0.06837705, …, 0.01078166],
        "parrots": [-4.03498672e-02, …, -3.69113460e-02],
        "can": [-0.08220862, …, -0.02696454],
        "live": [-0.10356505, …, 0.02606093],
        ...
}
```

```
Word Embedding Matrix:
[
        [-0.06837705, …, 0.01078166],
        [-4.03498672e-02, …, -3.69113460e-02],
        [-0.08220862, …, -0.02696454],
        [-0.10356505, …, 0.02606093],
        ...
]
```

5 Literature Review

Our work fits into a growing body of research and analysis of the Reddit community. Other students have applied deep learning methods to predict the "karma" (upvotes/downvotes) of a given Reddit comment [10], and there have been many attempts to understand user behavior through the topics posted about, as well as through the interactions that unfold in the comments section [8][3].

Deep learning approaches have been successful in other text-based problems, such as sentiment analysis [9], document labeling [1], and machine translation [2]. The benefits of these newer neural network approaches over more simple classifiers led us to explore ways to apply them to our task.

6 Results and Error Analysis

After fine-tuning hyper-parameters and applying masking to the RNN, we saw results that reflected an improvement of **2.96%** over bag-of-words (As seen in Table 3).

Subreddit	**Precision**	**Recall**	**f1-score**	**support**
aww	0.46	0.75	0.57	7405
technology	0.69	0.58	0.63	7457
movies	0.70	0.61	0.66	7395
Bitcoin	0.68	0.62	0.65	7579
gaming	0.57	0.54	0.56	7542
politics	0.74	0.71	0.72	7593
Music	0.72	0.63	0.67	7529
avg / total	0.65	0.63	0.64	52500

Table 3: n = 25000, Word2Vec (2.6 mil comments), RNN. Accuracy = **63.38%**

Upon looking at examples of incorrect labeling, we found several patterns. The first was the occurrence of deleted comments, appearing in the database as ”[deleted].” These comments slipped through the pre-processing step, and likely comprise at least 1% of the misclassified comments. For purposes of comparing the RNN against the bag-of-word’s relative performance, however, this is negligible as both are affected. We also observed that the subreddits which performed the best, such as ”politics” and ”Music,” remain the ones that seem to have the most keywords unique to their comments as we found during the Oracle.

We ultimately hypothesize that the RNN’s increase in performance over bag-of-words is due to its ability to capture sentiment and context, resulting in increased accuracy distributed across all subreddits.

References

[1] Mark J Berger. *Large Scale Multi-label Text Classification with Semantic Word Vectors*. URL: https://cs224d.stanford.edu/reports/BergerMark.pdf.

[2] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: (June 2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

[3] Daejin Choi et al. "Characterizing Conversation Patterns in Reddit". In: *Proceedings of the 2015 ACM on Conference on Online Social Networks - COSN '15*. New York, New York, USA: ACM Press, Nov. 2015, pp. 233–243. ISBN: 9781450339513. DOI: 10.1145/2817946.2817959. URL: http://dl.acm.org/citation.cfm?id=2817946.2817959.

[4] *I have every publicly available Reddit comment for research. 1.7 billion comments @ 250 GB compressed. Any interest in this?* URL: https://www.reddit.com/r/datasets/comments/3bxlg7/i%7B%5C_%7Dhave%7B%5C_%7Devery%7B%5C_%7Dpublicly%7B%5C_%7Davailable%7B%5C_%7Dreddit%7B%5C_%7Dcomment/ (visited on 12/11/2015).

[5] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. "An Empirical Exploration of Recurrent Network Architectures". In: *Proceedings of The 32nd International Conference on Machine Learning* 37 (2015), pp. 2342–2350. URL: http://jmlr.org/proceedings/papers/v37/jozefowicz15.html.

[6] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Nips* (2013), pp. 1–9. ISSN: 10495258. DOI: 10.1162/jmlr.2003.3.4-5.951. arXiv: 1310.4546.

[7] *Reddit Comments*. URL: https://www.kaggle.com/c/reddit-comments-may-2015 (visited on 12/11/2015).

[8] Philipp Singer et al. "Evolution of reddit: from the front page of the internet to a self-referential community?" In: (Apr. 2014), pp. 517–522. DOI: 10.1145/2567948.2576943. URL: http://dl.acm.org/citation.cfm?id=2567948.2576943.

[9] Duyu Tang, Bing Qin, and Ting Liu. "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* September (2015), pp. 1422–1432. URL: http://aclweb.org/anthology/D15-1167.

[10] Jason Ting. *A Look Into the World of Reddit with Neural Networks*. URL: https://cs224d.stanford.edu/reports/TingJason.pdf.

[11] *word2vec*. URL: https://code.google.com/p/word2vec/ (visited on 12/11/2015).