Machine Learning Nanodegree Capstone Proposal

Keenen Cates

February 2018

1 Domain Background

Getting a machine to understand the meaning of language is a largely important goal to a wide variety of fields, from advertising to entertainment. The branch of computing that aims to solve this problem of understanding is natural language processing, with sub-fields such as sentiment analysis. Sentiment analysis involves trying to understand the underlying sentiment and emotion behind language. For example, "Have a great day" has a positive sentiment, and "Have a bad day" has a negative sentiment. Currently techniques for modelling sentiment in language, involve using machine learning and deep neural networks to classify the sentiment of language. For example, SemEval is a yearly contest for trying to classify tweets as Positive, Negative, or Neutral. Its findings advance the field of sentiment analysis and machine learning [2].

My focus is on another major social platform, Youtube, which garners hundreds of thousands of comments and other user generated statistics. User data yields important results in the fields of social sciences. In particular I am interested in the top two-hundred trending youtube videos, and aim to identify sentiment of commenters by assigning an emoticon. Emoticons give insight into the comment writers sentiment. Using the subset of comments with emoticons I hope to develop a model that can identify what emoticon might accompany a comment to express sentiment [1].

1.1 Motivation

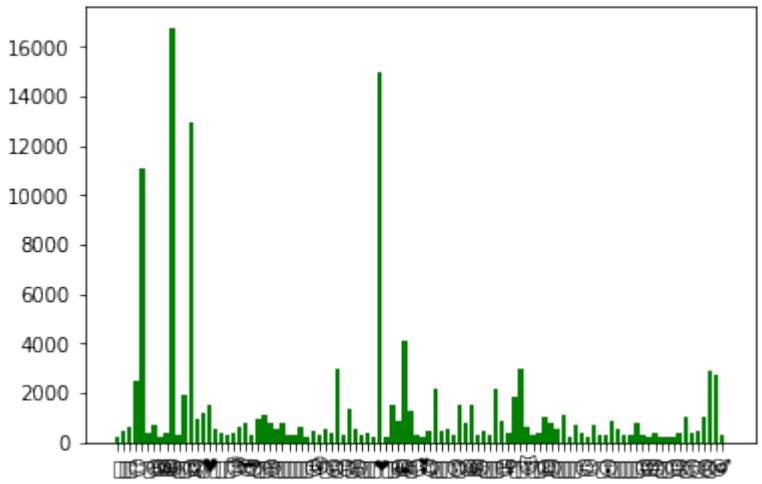
My motivation comes from the fact that semantics are hard in English. Conveying a sentiment is not an easy tasks in day to day conversation; however, emoticons allow us greater expressiveness in written communication via a pictographic language. For this reason, it would be interesting to see if one can predict emoticons a user might use with a text and even translate complex ideas into strings of emoticons that tell stories. Emoticons have the potential of removing some confounding factors of written language, as Emoticons can easily convey concepts, which would potentially remove difficulty in linking syntax to semantics.

2 Problem Statement

In this project, I will use deep learning, and in particular recurrent neural networks with LSTM layers in order to recommend what emoticons might accompany a Youtube Comment. Emoticons being pictographic symbols which can be used in text to convey concepts such as a smiling face, a sad face, and etc. There are many types of emoticons, from faces to common object and more are added to Unicode consistently. There are currently one-thousand forty-four emoticons that can be represented by Unicode [7]. A fully implemented model should be able to take new comments and return what emoticons it recommends a user to accompany this new text; based on the training of the model. My initial goal will be to unroll complex multi-label data points (comments with multiple emoticons) into multi-class data points by unraveling each label into multiple data points. This approach will not be the best way to approach multi-label classification; however, it is fairly simple and provides a landmark that can be expanded later as a stretch goal when I learn how approach such a problem.

3 Datasets and Inputs

I obtained comments from two-hundred top trending Youtube videos via a dataset on Kaggle. In order to create a labelled dataset, I will take a subset of these comments containing emoticons, and then extract those emoticons as labels. The unlabelled data will be kept purely for use in testing how the model preformed. Furthermore ten-percent of the labelled data is kept for validation purposes only, as a way of testing all model on the same data for ease of comparison. In total, there are 691, 388 rows in the dataset. A large proportion of them contain emoticons, (more than 200, 000), so there is a quite a bit of data, and it would be fairly straightforward to access the Youtube API and get more if needed. This means I have as much data as I could possibly want, and more if needed. As for features, I will only use the text, likes, reply threads, and so on will be ignored in this phase of the project. On average, each text is 15 words long. The Emoticons are not uniformly distributed either, leading to a largely unbalanced distribution. In fact, the crying happy face makes up a large portion of the data [8]. The graph below displays the skew of the distribution of emoticons.



4 Solution Statement

In order to tackle this problem, I will use a Reccurrent Neural Network with LSTM Cells as my main model to recommend data. I have chosen this type of model because of the previous successes; however there is no guarantee that this will preform well. Recurrent neural network with LSTMs are able to model long term patterns in text quite well; however, Youtube comments tend to be quite short and might not make use of the RNN models temporal element as much as a longer text. In addition, I will be modelling on the word-level, I think N-gram models might be worth exploring aswell if the accuracy is leaving something to be desired.

5 Benchmark Model

For comparison purposes, I will implement a Naive Bayes Model which some assumptions of independence in the data. In addition to the Bayesian model, a dummy model that simply predicts the most probable emoticons across all comments will be created. This is similar to an r-squared score in that I will use this to see how well I do compared to the "mean."

6 Evaluation Metrics

The models will be evaluated using a holdout set of data, in which each will recommend five emoticons that might accompany a text. If at least one recommendation is an emoticons that occurs in the validation comments, then I will consider it to be a "correct" guess. Accuracy is then the number of correct guesses divided by total guesses. Keras calls this accuracy "top k

categorical accuracy". Mathematically, this would look something like this where matching $x \in Comments$ and $y \in Labels$ and $score(x) = 1$ if any $p \in argmax_{k=5}(predict\_labels(x))$ is in y, else $score(x) = 0$. $predict\_labels(x)$ would return the probabilities of each output class occurring. Then the accuracy of the model would be $\frac{\sum_{i=0}^{N}(score(x_i))}{N}$ where $x_i \in Comments$ and $N = \mid Comments \mid$

However, the unbalanced nature of the distribution of emoticons means that I will likely need to use F1 score as an evaluation metric. I am not sure which one will be best in practice yet, but I believe that F1 will likely be critical to evaluating the model.

7 Project Design

7.1 Programming Language Libraries

- **Python 3**
- **TFLearn** a deep learning library featuring a higher-level API for TensorFlow.
- **TensorFlow** a deep learning library

7.2 Machine Learning Design

- **Training:** Using engineered labelled data-set. What I mean by this, is that the original text has no labels; however, I will extract the emoticons out as a "feature" that I will attempt to predict. The feature will not take into account multiple occurrences of the same emoticon; therefore the label will be the set of emoticons that appeared.. Multi-class labels will need to be unrolled into multiple entries.
- **Representation:** Recurrent Neural Network with word2vec model
- **Architecture:** Deep Recurrent Network with multiple layers of Long Term Short Memory cells stacked

7.3 Additional Algorithms

- **Naive Bayes:** A simplistic model that can be used to great effect at times. Based on Bayesian Statistics i.e. Bayes Rule, and will provide an easy to implement model given that I make some assumptions about that data's independence. For its simplicity it will serve as an excellent baseline model.

7.4 Plan of Attack

- First order of business will be to straighten out the data. This means preforming all necessary data cleaning i.e. extracting emoticons as labels. Then doing tasks such as converting text to lower case, converting punctuation to tokens, embedding the words if desired, and one-hot encoding of the labels (for multi-class). Then after pre-processing, saving the data in a way that is useful. At this stage, it will likely be necessary to create some sort of holdout set for validation across models. More time spent here will likely improve the models, and might be an opportune time to figure out how to approach multi-label classification.

- Next, implementation of the Bayesian will be critical to assessing the performance of the Deep Learning model. For this reason, I will implement this model first and this will allow me to find any problems that might arise from the cleaning process before approaching the more complicated RNN model.

- From this point on, I will attempt to create a model using Deep learning with TFLearn. I think this part should be fairly straight forward given previous experience in class with RNNs with LSTM cells. More time might need to be spent getting the data in order, i.e. fixing the length of the comments. Optimizing the model might pose a problem, as there is likely many improvements I might make to the architecture (i.e. batch normalization or dropout); however, I'm not sure yet what will be the best method.

- Lastly, I will create functions the make the actual prediction given a piece of text, and assess the accuracy of each model.

8 References

1. Hogenboom, Alexander, et al. "Exploiting emoticons in sentiment analysis." Proceedings of the 28th Annual ACM Symposium on Applied Computing. ACM, 2013.

2. Rosenthal, Sara, Noura Farra, and Preslav Nakov. "SemEval-2017 task 4: Sentiment analysis in Twitter." Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). 2017.

3. Salas-Zárate, M. P., Medina-Moreira, J., Lagos-Ortiz, K., Luna-Aveiga, H., Rodríguez-García, M. Á., & Valencia-García, R. "Sentiment Analysis on Tweets about Diabetes: An Aspect-Level Approach." Computational & Mathematical Methods In Medicine, 1-9. 2017.

4. Siersdorfer, Stefan, et al. "How useful are your comments?: analyzing and predicting youtube comments and comment ratings." Proceedings of the 19th international conference on World wide web. ACM, 2010.

5. Taboada, Maite, et al. "Lexicon-based methods for sentiment analysis." Computational linguistics 37.2 (2011): 267-307. Kang, Mangi, Jaelim Ahn, and Kichun Lee. "Opinion mining using ensemble text hidden Markov models for text classification." Expert Systems with Applications (2017).

6. Lee, Ji Young, and Franck Dernoncourt. "Sequential short-text classification with recurrent and convolutional neural networks." arXiv preprint arXiv:1603.03827 (2016).

7. "Emoji." Wikipedia, Wikimedia Foundation, 7 Feb. 2018, en.wikipedia.org/wiki/Emoji.

8. Mitchell, J. (Datasnaek). "Trending YouTube Video Statistics and Comments." Kaggle, Kaggle Inc., Aug./Sep. 2017. https://www.kaggle.com/datasnaek/youtube/