COMP 6630-001: Project Proposal

Matthew Freestone*
Auburn University
United States of America
maf0083@auburn.edu

Will Humphlett
Auburn University
United States of America
wh@auburn.edu

Matthew Shiplett
Auburn University
United States of America
mss0033@auburn.edu

ABSTRACT

This project proposal will focus on an outline for using a multi-layer perceptron to perform sentiment analysis on movie reviews. The purpose of this model is to create an automated function capable of English text sentiment analysis. After training on a dataset of labeled IMDB movie reviews, the model will be able to determine if review's text sentiment is generally positive or negative. The model's accuracy will be confirmed by testing on text that is not about movies. Once trained, this model could be used by various entities to expedite sentiment analysis on large bodies of English text based feedback. Examples include corporate entities running analysis on feedback of their products is mostly positive or negative, or instructors running analysis on feedback of their courses.

CCS CONCEPTS

• **Theory of computation → Machine learning theory**.

KEYWORDS

multi-layer perceptrons, machine learning, sentiment analysis, natural-language processing

ACM Reference Format:
Matthew Freestone, Will Humphlett, and Matthew Shiplett. 2022. COMP 6630-001: Project Proposal. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 PROPOSAL

- Define clearly what problem will you work on. (What is the primary application domain?
  - This project will aim to use a multi-layer perceptron to classify English text based on sentiment. The primary domain is on English text, with inputs being a dictionary of words, represented by a 1D vector, containing integer counts of the number of times a word in the dictionary appears within a selected body of text. The output being a number {-1,0, 1} representing a sentiment of {negative, neutral, positive} respectively.
- How MLP can be applied to solve this problem?
  - Multi-layer Perceptrons (MLP) can be used to accomplish this classification task. With the use of multiple layers, we are able to achieve highly non-linear decision planes, allowing the model to separate text into categories of negative, neutral, and positive. The perceptrons will be fully connected at each layer, and each will learn weights based on the training data. Back-propagation will be used to update the weights at each level, and Stochastic Gradient descent will be used to converge upon a model with high accuracy.
  - In order to achieve the 3-class output, we have two possible implementations to try. We could create a single MLP with 3 classes { Positive, Negative, Neutral }. We could also create a two-stage system, with a first MLP making a binary decision between { Neutral, Some Sentiment }, and if that MLP determines that there was a sentiment, a second MLP will make a choice between {Positive, Negative}.
- Briefly identify who the users of your classifier will be, and why it is important to implement this classifier.
  - Sentiment analysis is a valuable tool for data scientists attempting to analyze text. Ideally, the model will be able to quickly classify large amounts of data reasonably accurately, providing the users with a way to determine the sentiment of text. Additionally, corporate entities looking to perform sentiment analysis on a large body of feedback can use such a model to quickly gain insight into the sentiment of the feedback.
- What are the potential challenges you may face in this project?
  - Due to the large dictionary of words found in English, even a limited subset, the MLP will need to have inputs equal to the dictionary size. Even for a small network, the number of connections will be large in order to accommodate the large input space. The large size of the network can mean training and backpropagation will be computationally intensive. Compounding these issues, given our input mode the order of words will be disregarded. The proposed dictionary vector will only model the frequency of appearance of words, but not the order in which they appear.
  - Determining how to pre-process text for the model will be challenging, as there are a large variety of methods to do this. We plan to try a number of them, including TF-IDF, N-Grams, and Word2Vec.
- What is your dataset? How will you create/build your dataset?
  - In order to have a well-defined and strongly separable dataset, we plan to use a labeled dataset from Kaggle containing 50,000 movie reviews from IMDB. The data points are labeled with "positive" and "negative" corresponding to the sentiment of the text. The description of the dataset describes the points as "highly polar," meaning that we are likely to find a model that separates the data well. If the project requires more data than 25,000 for training in the Kaggle dataset, we plan to perform data scraping on IMDB's website. We can determine the label for the points with the movie ranking associated with the review. We could also retrieve neutral points from this method, perhaps by marking reviews with between a 5 and a 7 as neutral. The dataset we will use initially can be found

*Project Coordinator

here: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
  - As an addition to the IMDB reviews dataset, we plan to make use of a dataset that consists of 1.5 million tweets labeled with POSITIVE, NEGATIVE, NEUTRAL sentiment. This dataset will allow us to augment the training of our model to detect NEUTRAL sentiment. Detecting a neutral sentient will benefit the model insofar as not all potential input text will necessarily have a strong, meaningful positive or negative sentiment. Therefore, it would be undesirable to have the model learn that all text must be labeled as either POSITIVE or NEGATIVE as that false dichotomy could lead to the misclassification of text samples from outside the IMDB Highly Polarized data.
- How do you plan to build it? Identify what technologies you plan on leveraging to implement your software. This may be programming languages, supporting libraries, etc.
  - The plan is to implement the bulk of the model's code in python. The code repository and version control will be managed using GitHub. We will use Anaconda for package management, and jupyter notebooks develop code in. Supporting libraries will likely include Numpy, Pandas, and for later comparison libraries such as SkLearn and Pytorch may be used to instantiate comparative models.
- What hyper-parameters will be involved in your classifier? How would you fine-tune these hyper-parameters?
  - As the classifier will be a Multi-layer perceptron, we will need to determine the right number and size of layers in the model. This hyperparameter could have many different values, and it might be difficult to tune. We also will need to determine optimal values for learning rate, number of epochs, and activation function(s). To determine optimal hyperparameter values, we will utilize a parameter matrix that covers the search space and identify the set that produces the highest accuracy without overfitting. The dataset is also small enough to allow us to test the accuracy difference between Gradient Descent and Stochastic Gradient Descent.
- How will you demonstrate the usefulness of your classifier?
  - The classifiers purpose is to perform sentiment analysis, and given the it is a machine classifier it should be able to perform sentiment analysis significantly faster than a human. Therefore, to demonstrate usefulness, the model will be applied to a large body of previously uncategorized text, producing sentiment labels significantly faster than a human could categorize the same body of text.
  - In order to determine how well our model performs against others, we will create use models, like SVM, Random Forest, and Naive Bayes, and train them on the same dataset. We will use several metrics to determine how our model holds up, including Accuracy, Precision, Recall, and F1 Score. We will also consider using Mean Reciprocal Rank, if it shows more meaningful results.
- Provide a rough timeline to show when you expect to finish what. List a couple of milestones if possible (they can be tentative).
  - Project Proposal - Due : October 26th, 2022
  - Initial Commit of Repository - Expected : October 26th, 2022
  - Proposal Feedback - Expected : After October 26th, 2022
  - Dataset acquisition and pre-processing - Expected November 7th, 2022
  - Initial implementation of MLP - Expected : November 9th, 2022
  - Initial training run of MLP - Expected : November 10th, 2022
  - Verification and recording of results - Expected : November 11th, 2022
  - Initialization of Benchmark implementation(s) - Expected November 11th, 2022
  - Verification and recording of benchmark results - Expected November 15th, 2022
  - Creation of Final report documentation - Expected November 16th, 2022
  - Finalization of Final report documentation - Expected November 18th, 2022
  - Rehearsals of Final report - Expected November 28th - December 1st, 2022
  - Project presentation - Expected December 1st, 2022