Udacity Machine Learning Engineer Nanodegree

Capstone Project Proposal

Juan Andrés Mora

March 1st, 2021

Domain Background

Machine Learning has been applied to a wide variety of problems. One of the most popular areas of study is Natural Language Processing. Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data[1]. In simpler terms, it allows a computer to understand a person.

On the other hand, one of the most fast and complete sources of information is social media. People everywhere use Twitter and report on real time any emergency they may be observing. But there is no specific signal for warning about a disaster. Therefore, a Natural Language Processing may be used to detect if a user is writing about a disaster or not.

This is a particularly exciting problem for me as I see it as a basis for a direct application useful in my city. During October 2019, our society lived through a period that was later denominated “social unrest”. During this month there were many points throughout the city in which riots and protests became violent and dangerous. There existed no official information about this, and news media could not keep track of all the spontaneously dangerous locations in the city. Therefore, Twitter was the most useful platform to check whether it was safe to follow a specific route or visit a given place as people constantly reported points of conflict between protesters and the police. An algorithm to quickly detect if someone is reporting a dangerous location could help prevent people accidentally walking into a conflict point and potentially be injured.

[1] “Natural Language Processing”, Wikipedia, https://en.wikipedia.org/wiki/Natural_language_processing

Problem Statement

My selected problem is "Natural Language Processing with Disaster Tweets" an ongoing Kaggle competition[2]. The objective is to predict which Tweets are about real disasters and which ones are not.

Datasets and Inputs

The dataset can be found in the official Kaggle competition webpage[3]. It contains three files: a training set (train.csv), a test set (test.csv) and a sample submission file for the competition (sample_submission.csv). The data provided has several columns:

- id: unique identifier for the tweet.
- text: the text of the tweet.
- location: location the tweet was sent from.
- keyword: particular keyword from the tweet.
- target: only available in the train set, defines a disaster tweet (1) or not (0).

The training set contains 7614 rows (including a header row) and the test set contains 3264 rows (also including a header).

As only the train test has the target column information, we will be splitting this data in order to use part of it to train the model and the remaining data will be used to evaluate the model's performance.

Solution Statement

The proposed solution will consist of several steps. We will first prepare and process the data, cleaning the text from irrelevant information and padding the text to a homogeneous structure. The next step will be to implement a recurrent neural network (RNN) and train it using the methodology described in this course. The model will then be used to predict if the Tweet refers to a disaster or not on a different portion of data. Then, the model predictions will be evaluated and compared to a benchmark model.

Benchmark Model

As a benchmark model, we will also implement the basic linear Ridge Classifier described in the official tutorial[4]. This is a simple model but will be useful as a starting point and will help show to what extent our neural network improves predictions.

---

[2] https://www.kaggle.com/c/nlp-getting-started/overview
[3] https://www.kaggle.com/c/nlp-getting-started/data
[4] https://www.kaggle.com/philculliton/nlp-getting-started-tutorial

Evaluation Metrics

As described in the Kaggle evaluation section[5], the competition uses F1 between the predicted and the expected answers. It is calculated as:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Where:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

TP: True Positive = prediction is 1 and ground truth is 1

FP: False Positive = prediction is 1 and ground truth is 0

FN: False Negative = prediction is 0 and ground truth is 1

The benchmark model in the tutorial uses cross-validation and reports an F1 score between 57% and 64%. This will then be the metric our model will try to improve on.

Project Design

The intended workflow is the following:

1. Data exploration: examine the provided dataset and check the data distribution.
2. Dataset generation: from the labeled file provided generate a training set and a test set.
3. Prepare and process the data: clean the text provided (ex: remove html tags, emojis, stopwords, etc.), stem and split into words.
4. Transform the data: generate a dictionary, convert the data to a numeric notation and pad the text to have homogeneous length.
5. Benchmark model: build and train the basic model proposed as benchmark.
6. RNN model: build and train our more complex model.
7. Evaluation: compare performance of both models on the test set.

[5] https://www.kaggle.com/c/nlp-getting-started/overview/evaluation