AI and NLP in Historical Document Analysis

Kellen Caron
kcaro419@mtroyal.ca

Eric Chalmers & Jordan Kidney

Department of Mathematics and Computing
Mount Royal University

Abstract. This paper outlines the development of a Natural Language Processing (NLP) model training system aimed at performing sentiment analysis on declassified Cold War era CIA documents. Limitations lead to the full scope being unrealized, but significant progress was still made in the development of the NLP model training system. Feature engineering via Term Frequency – Inverse Document Frequency (TF-IDF), and classification via Multinomial Logistic Regression (MLR), cross-entropy loss and gradient descent were utilized to categorize documents across six distinct emotions. Challenges arose in aligning probabilities across terms into documents, and again in aligning loss values across documents into terms. The complexities encountered in aligning probabilities and loss values demonstrate the need for further investigation and refinement, and affirm the potential of the system. While the model training system is just shy of completion, current training attempts show promising trends, and suggest success with additional refinement. Future work involves comprehensive training on a complete data set and rigorous testing in order to validate full functionality and accuracy of the system.

Keywords: Natural Language Processing, Term Frequency – Inverse Document Frequency.

1 Introduction and Project Overview

1.1 Original Plan of Approach

At present, the title of this paper is, unfortunately, misleading. The original scope of this project was to leverage Artificial Intelligence (AI) and Machine Learning (ML), with Natural Language Processing (NLP) techniques in concert with declassified Central Intelligence Agency (CIA) documents from the Cold War era in order to analyze patters, relationships, and shifts in alliances and collaborations to provide insight into the dynamics of intelligence operations during this period. This was to be done via the design and development of an NLP model training system tailored towards analysis, all of which would be coded in C#, and the subsequent utilization of that NLP model in conjunction with some traditional historical analysis for review. The ultimate goal of this being the expansion of personal ability in the field of AI and ML, alongside practice in the usage of the C# language. Unfortunately, the initial scope proved to be too broad to fit within the time constraints present. Instead, this project was fulfilled up to the completion of the training system, and prior to its full testing. As such, the contents of this paper will predominantly be about, and focus on, the development of this system; how it was achieved, what steps were taken to build

it, and the rationale behind certain development choices. To begin with, let us first cover some of the core concepts utilized throughout development; it is worth noting, while we do, that much of the research done for this paper was conducted with the aid of chatGPT, with its output being verified through other, proper sources.

1.2 NLP explanation and overview

Natural Language Processing, or NLP, techniques were at the core of this project, representing an important methodological approach in analyzing the utilized data set. NLP is a branch of artificial intelligence that focuses on creating tools and systems to understand, interpret, and generate human language [1, p.448-9]. Going deeper, this project further focused in on the Sentiment Analysis branch of NLP, also called opinion mining, a branch focused in on analyzing 'sentiment'; that is, on interpreting some sort of emotional, sentimental context of some given text [2, p.12]. This specific subset of AI/ML work was chosen due to its natural connection with the subject matter at hand; getting a machine to make sense of text naturally falls under the NLP umbrella. Sentiment analysis was further focused in on, as opposed to some other branch of analysis, due to a perceived broad applicability that made sentiment analysis seem like an approach that would not only allow for interpretation of the target documentation, but would also allow for interpretation of a broad range of other text sources should events have forced a pivot in this project's target of analysis. Finally, this approach was also guided by my own background in the study of history, something I believed at the time of starting would prove useful in verifying the given results of AI/ML analysis. With this in mind, we can now talk about the NLP model training system that was ultimately developed for this project, and the details of its development.

In designing this NLP model training system, the first step to handle was the acquisition of data to train with. In this instance, the construction and curation of a specifically tailored data set was seen as far too costly an endeavor, in terms of time, to be feasible for this project, so instead a pre-made data set was found and used. The data set in question was one developed by Elvis Saravia, Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen, for their paper *Contextualized Affect Representations for Emotion Recognition*, in '*Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*' [3]. This data set featured some 20,000 tweets, each tied to a corresponding emotion, one of love, fear, joy, sadness, surprise, and anger. In this way, a model trained on this data to perform sentiment analysis would be able to categorize given documents into one of those six emotions, ideally expanding the potential document base to which the model would be usefully applicable. Another step taken to help ensure a functioning training system was to, at each step, save the output of any given calculation into a JSON document; in this way, the process could be completed in distinct stages, with a way to visually verify output at each stage. With this data set in hand, the next portion of this project was to tackle feature engineering, specifically using the Term Frequency – Inverse Document Frequency technique.

2 Feature Engineering

2.1 Overview

Feature engineering refers to the general process of creating numeric representations out of our raw data; that is, taking our human readable data, and transforming it into something machine-friendly [4, p.15]. Naturally, this is the first necessary step in having a machine work with our data, and as we embark on it we must first engage in some data preprocessing, to clean and prepare our data for the later steps. Once our preprocessing is done, we then move on to employing specific techniques of feature engineering; in this instance, the Term Frequency – Inverse Document Frequency (TF-IDF) technique.

2.2 Data Preprocessing

The first step of preprocessing was developing a means of splitting apart the larger dataset into chunks that would be both easier and faster to work with for testing, as well as allow for the dataset to be split into a final set for trainer purposes, and a set for validation purposes. With the dataset appropriately split apart, the next step is the removal of stop words; common english terms that lack significant meaning, such as 'the' or 'if'. As the premade dataset being used has already removed things like punctuation, while organizing the data into an immediately usable format, the data preprocessing step here is composed of removing unimportant stop words to produce a corpus, and recording every unique term present to produce a vocabulary. For the purposes of removing stop words, another premade list was utilized; this time, the Natural Language Toolkit's (NLTK) list of english stop words was utilized as an initial starting point [5]. However, this list of stop words was also added to; every word for which there was only a single instance across the entire dataset was added appended to the stop word list, leaving us with 35,870 total stop words removed from our dataset. Once stop words have been removed from the dataset we can call what remains our corpus, a collection of documents ready for further interpretation. The last stage of preprocessing is simply to record every unique occurrence of a term in order to build a vocabulary of used terms across all documents in the dataset.

2.3 Term Frequency – Inverse Document Frequency

TF-IDF is the specific feature engineering technique that was employed for this NLP model training system. TF-IDF is a technique that is meant to provide a numerical measure of the importance of any given term inside a document, but with reference to that terms importance across the dataset as a whole [6, p.7-8]. This technique was chosen given its relative simplicity to employ, an important feature given the various constraints upon this project. Term frequency is, as the name implies, a measurement of how frequently a term appears inside a given document; it is simply the number of times a term appears inside a document [6, p.8-9]. As the documents utilized for this project were made up of tweets, and so exceptionally short, the term frequency was also normalized by dividing it by the total number of terms, not including stop words, inside that document, as expressed by the equation:

$$TF_{ij} = \frac{f_{ij}}{T_j} \tag{1}$$

Where we have a collection of N documents, with f_{ij} as the number of occurrences, or frequency, of a term i in document j, and T_j as the total number of terms in document j. Inverse document frequency refers to how we interpret the importance of that term across the entire dataset [6, p.8-9]. That is to say, we take the log of the ratio between the total number of documents N, and the number of documents n containing the term i, as expressed by the equation:

$$IDF_i = ln\left(\frac{N}{n_i}\right) \tag{2}$$

These two calculations are then combined via a simple multiplication across the document to produce TF-IDF scores for each term within each document. The next stage of this NLP model training system is classification.

3 Classification

3.1 Overview

Classification is the next step, and involves the application of predictive modeling algorithms. Classification here refers to the process of having the model learn from the labeled and classified data prepared through feature engineering, to be able to label and classify new and unseen data [7, p.221-2]. For this project, as the data was already labeled and classified based on six emotions, the goal is thusly to train our NLP model to classify new data based on one of those six emotions, our classes. To do this training, a predictive modeling algorithm is needed; in this case, Multinomial Logistic Regression (MLR).

3.2 Multinomial Logistic Regression

MLR is an algorithm that expands upon Logistic Regression (LR), one of the simpler linear classifiers; this relative simplicity being the core reason it was selected for use. MLR uses multiple categories or classes, for this project the 6 emotions, each of which is mutually exclusive; the probability of a given document belonging to each category is calculated using predictor variables, in this case a set of weights and biases [8, p.15]. The weights used here are numbers attached to each term in our vocabulary, and categorized under each emotion; that is, all terms possess a set of weights indicating how heavily they are 'weighted' towards each emotion. The biases involved are attached to each class alone, and are a number representing the 'bias' that should be given to a class when considering which class a document belongs under. At initialization, all bias and weight values were set at some random value extremely close to 0, to avoid symmetry that may have occurred from beginning at 0 for everything, and to ensure that a later calculation would not return 0 by default. From here, the remaining implementation of MLR in this project can be broken down into stages; forward propagation and the softmax function, the cost function, and gradient descent.

Forward Propagation and the Softmax Function. Forward propagation refers to the process of inputting some data x to receive an output y such that the initial information from x propagates through various transforming layers to produce y [1, p.197-8]. In

this project, the forward propagation step meant working through each weight, factoring in TF-IDF values, then adding in the relevant bias value to produce a final class score. That is, if we have some W_{ij} representing the weights W at emotion or class i and term j, some bi representing the bias value of that emotion, and some x_{ij} representing the TF-IDF values, then we can express the resulting class score value S_{ij} with the following equation:

$$S_{ij} = \sum_{i}\left(W_{ij} \times x_{ij} + b_{i}\right) \tag{3}$$

This resulting value is then fed into a softmax function, whose role is to compute the class scores into probabilities by ensuring that, when all class scores attached to a term are added together, they sum to 1. That is, we can find the resulting probability P_{ij} through the following equation:

$$P_{ij} = \frac{e^{S_{ij}}}{\sum_{i} e^{S_{ij}}} \tag{4}$$

However, this method produced another problem that needed to be solved. Namely, these probabilities were calculated for each term, but it is the ultimate goal of this stage to possess probabilities for each document.

The solution to acquiring probabilities at the document, as opposed to the term, level was simply to match terms to those documents which contained them, and then combine the term percentages with equal weighting. Equal weighting was chosen in this instance due to the inclusion of TF-IDF scores at the prior step; the probability values for each term were already weighted by importance, making further efforts here to weight their importance redundant, if not inaccurate. As such, once each document within the corpus had a set of probabilities attached to it for each term contained within, the probabilities of each term in that corpus were simply combined, and then re-normalized to appear as proper percentages.

Cost Function – Cross Entropy Loss. The cost function is the next stage, and refers to a measure that evaluates how well the predictions made at the forward propagation step align with reality [1, p.172-3]. However, cost function is also a general term, with the specific kind of cost function utilized for this project being cross-entropy loss. Cross-entropy loss measures the dissimilarity between the probability predicted, and the actual value, to produce a loss score for each document within each class; the ultimate goal being to use these loss scores to alter the weights and biases such that subsequently calculated probabilities are closer to reality [1, p.215]. Cross-entropy loss was chosen for this project due to ease of use, effectiveness in producing accurate probability distributions, and recommendation from project supervisors. For the implementation of cross-entropy loss here, let L_{ij} be the loss score for emotion or class i in document j, $\hat{y}_{ij}$ be the predicted probability, y_{ij} be the real value, in this case 1 for true and 0 for false, and let ε be some small value, here $1x10^{-15}$, added to avoid taking the logarithm of zero or one, so we can express our loss score with the equation:

$$L_{ij} = -\, y_{ij} \cdot log\left(max\left(\varepsilon, min\left(1 - \varepsilon, \hat{y}_{ij}\right)\right)\right) \tag{5}$$

However, this leaves a similar problem to that of the forward propagation and softmax step, as the resulting loss scores are attached to documents, but weights are attached to terms; meaning that the results of this equation must be split among the involved terms in some way.

The solution at this stage is slightly more complex than that utilized when producing document probabilities in the forward propagation and softmax stage. Splitting the loss score among a document's constituent terms equally would produce skewed results, as not all terms equally contribute to a document's probability, and therefore a document's loss. Instead, TF-IDF scores are once again utilized to weight the loss score as it is split among terms, ensuring that the loss value is parceled out in a manner consistent with term importance. To do this, we let a term's loss score be equal to the sum of the loss score of every document that term is present in, multiplied by the TF-IDF score of the term in that document. If we let T_{ij} be the term loss score for class or emotion i for term j, L_{dj} be the document loss score for term j within document d, and let $TFIDF_{ijd}$ be the TF-IDF value, then we can represent our term loss score with the equation:

$$T_{ij} = \sum_{d} L_{dj} \times TFIDF_{ijd} \tag{6}$$

This leaves only the final step, the gradient descent, wherein these loss values are utilized to adjust the weights and biases to improve performance in the following iterations.

Gradient Descent. Gradient descent is the stage at which the weights and biases are adjusted and updated in small steps so as to minimize loss values in future iterations. The gradient descent process involves computing the gradient, or derivative, of the cost function with respect to each model parameter; this gradient indicating the direction of the steepest ascent, meaning the negative of the gradient is towards the steepest descent and adjusting the parameters in that direction should thus reduce loss values [6, p.336,485]. This algorithm also features a learning rate, a value meant to determine the speed at which weights change with each iteration. This step is rather simple for this project, as it was mostly accomplished when each document's loss scores were divvied up among its terms; at that point, where the TF-IDF scores were applied to the term loss values, the gradients were effectively computed. As such, letting W_{ij} be the weight value for term j under emotion i, letting B_i be the bias value for emotion i, and letting R be the learning rate, the final equation can be represented as:

$$W_{ij} = W_{ij} - R \times T_{ij} \tag{7}$$

For the weights, and as:

$$B_i = B_i - R \times L_{dj} \tag{8}$$

For the biases. Biases receive a different treatment here, utilizing document loss values as opposed to term loss values, as they are meant as an aggregated measure, and so do not need to be, nor should be, influenced by TF-IDF scores in their updating. With this step completed, the only remaining work to be done is to run

through multiple iterations of classification, so as to train the NLP model, before utilizing it to gauge performance.

4 Outcome

4.1 Results

Unfortunately, given the time constraints of the project, only limited trained was accomplished which, while providing valuable data on the NLP model training system's potential, falls short of providing a full picture of function. That said, there is a means of basic evaluation available, in the form of reviewing average weight change per iteration. Since the goal of the MLR classification is to reduce the loss values to some optimal minimum, we can reasonably expect that around that point, the average change in weight values will level out. That is, while large changes around the first iterations are expected, as the training system accounts for larger loss values, as those loss values get smaller, average weight changes will get smaller and more consistent, as the optimal value is overshot in some direction by roughly equal amounts each iteration. In this way, by graphing average weight change vs. iteration, we can get a rough visual on the training system's ability; a hyperbolic curve indicating success. Figure 1 shows just such a graph, and demonstrates the desired shape to indicate success of the NLP training system. While this does not exactly indicate a functioning algorithm, it does suggest as much, and leaves hope that with some additional training on a full training data set, the successful function of the training system can be confirmed.

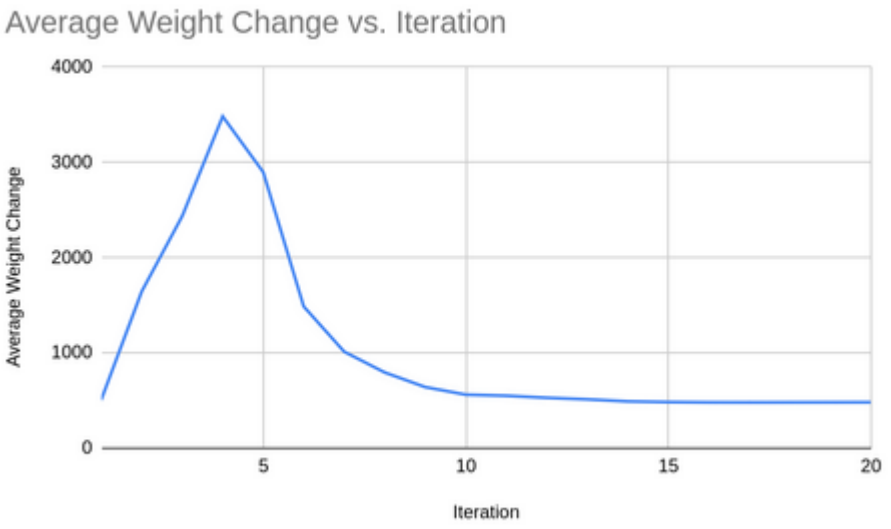

Fig. 1. Average weight change vs. iteration, using a data set composed of 10,000 tweets, and across 20 iterations.

4.2 Conclusion

In summary, this project was originally envisioned as developing an NLP model training system in order to perform sentiment analysis of declassified CIA documents from the Cold War era. While the initial scope proved too expansive given the constraints present, significant progress was still made in the development of the NLP model training system. Similarly, the core project goals of developing skills in the field of AI and ML, alongside skills in the C# language, were handily achieved. Pivotal steps in the development of this training system included feature engineering via TF-IDF, and classification through MLR and gradient descent for parameter adjustment. Each stage presented challenges and solutions, all contributing to a greater understanding of NLP techniques and sentiment analysis. While the limitations of the project leave knowledge of its actual performance limited, and further exploration and refinement is needed for completion, an analysis of average weight change over time suggests the system is on the right track. Moving forward, work would need to be done to complete training on a full data set, alongside the proper testing of such a produced model to ensure accuracy.

References

1. Goodfellow, I., Bengio, Y., and Courville, A. "Deep Learning." MIT Press, 2016.
2. Liu, B. "Sentiment Analysis and Opinion Mining." Morgan & Claypool Publishers, 2012.
3. Saravia, E., Liu, H.-C. T., Huang, Y.-H., Wu, J., Chen, Y.-S. "{CARER}: Contextualized Affect Representations for Emotion Recognition." In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Oct-Nov 2018, Brussels, Belgium. Association for Computational Linguistics, pp. 3687-3697. Available: https://www.aclweb.org/anthology/D18-1404 DOI: 10.18653/v1/D18-1404.
4. Zheng, A., Casari, A. "Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists." O'Reilly Media, 2018.
5. NLTK Project. (2010). English stopwords list. Retrieved from Gist GitHub Repository: https://gist.github.com/sebleier/554280#file-nltk-s-list-of-english-stopwords
6. Leskovec, J., Rajaraman, A., Ullman, J. D. "Mining of Massive Datasets." Cambridge University Press, 2014.
7. Bird, S., Klein, E., Loper, E. "Natural Language Processing with Python." O'Reilly Media, 2009.
8. Garson, G. D. "Logistic Regression: Binary and Multinomial." SAGE Publications, 2013.