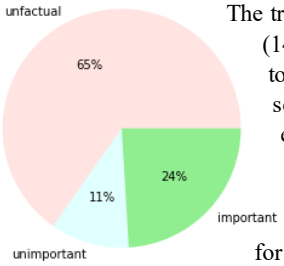
Introduction

The task of classifying text sentences into categories of factual importance is a pivotal challenge in the realm of Natural Language Processing (NLP) with far-reaching implications across various domains, including media, education, and information technology. At its core, this classification task aims to discern the veracity and significance of information, thereby enabling systems to filter out unfactual content, identify irrelevant data, and highlight critical facts. This is particularly crucial in today's digital age, where the exponential growth of information and the prevalence of misinformation demand sophisticated tools for ensuring the reliability and relevance of content that reaches users. In this assignment, I highlight the results of 3 machine learning models that we can use and the different methods of data preparation which worked best for them.

Data Analysis



The training file "train.csv" contains dataset distributed among 3 classes: unfactual data labelled -1 (14685), unimportant data labelled 0 (2403) and important data labelled 1 (5413). There was a total of 25k text sentences. The average number of words turned out to be about 17 words per sentence with its maximum having 152 words, while the lowest was a single word. Upon examining the data, it was found to be extremely unbalanced, with the unfactual dataset accounting for 65% of total training data as shown in Figure 1. I then try to identify the top 10 words in each class, this gives me an idea of which word makes up the most of the texts for each class as shown in Figure 2. It was observed that the word "the" is the most common word for all three classes. It is expected that stop words take up the top spots in Figure 2 due to their high frequencies.

Fig. 1: Distribution of "train.csv" before split

(a) (b) (c)

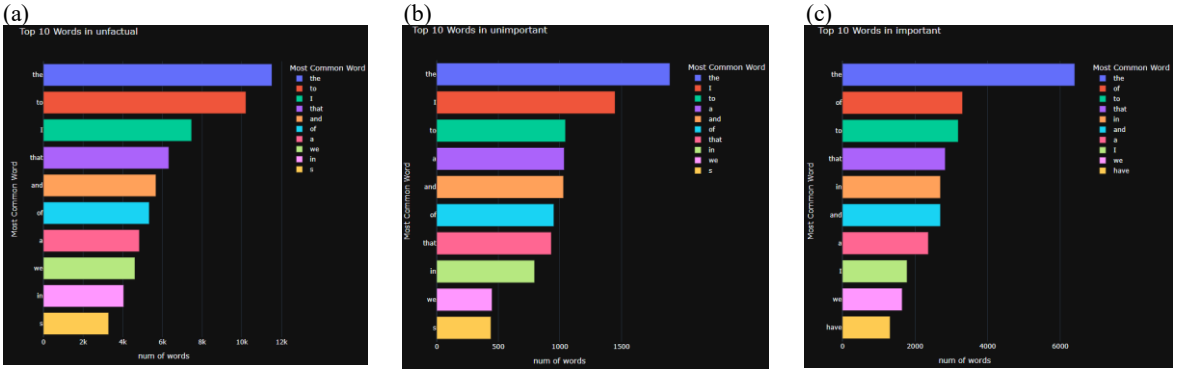


Fig. 2: Top 10 most common words in (a) unfactual, (b) unimportant and (c) important class

Data Cleaning and Augmentation

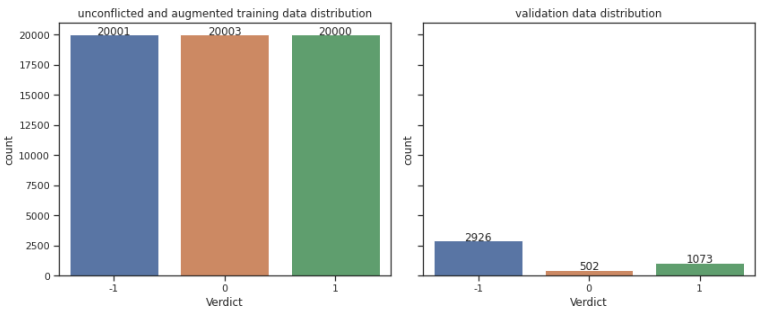


Fig. 3: Distribution of train and validation data after augmentation

While combing through the training data, it was observed that there were conflicting text sentences. These sentences are exactly the same but were labelled differently. For instance, the training data text sentence, "Here we are with a government that's been dealing with a drug-running Panamanian dictator." are repeated at least twice, and the labels for these repeated sentences were inconsistent. Sentences with repeated non-unique labels were then filtered out. To resolve the unbalanced classes, data augmentation was employed using TextAttack's Easy Data Augmenter (EDA). This method perturbs the original sentence in 4 ways: random insertion, deletion, synonym replacement or swap and is a well-known method in many text augmentation methods. Only the training data that had been split earlier from the original "train.csv" file were augmented. Figure 3 shows the final distribution data across the 3 classes in the training and validation set after the data augmentation. I have also tried other method of augmentation such as word embeddings, however other methods do not yield nearly as good results as EDA. In addition, word embeddings generate one augmented text with each call which makes it an extremely long and tiring process to generate data to even out the classes.

Data Preprocessing and Feature Engineering

Different methods and combinations of preprocessing were utilized. The text is first expanded of any contraction, converted to lower case and stripped of any newlines or return carriages. All special characters and punctuations were removed. The remaining words were then tokenized. I then set different parameters during my preprocessing step to enable an ablation study afterwards. The parameters were: including the stop words, including a set of exclusive stop words, including and short words (words that are less than 2 characters), and finally including and lemmatization or stemming processes. Exclusive stop words are stop words that contributes greatly to the overall sentiment of the text, and removing them could affect the overall sentiment of the data which then affects the factuality of the text sentence. In my first initial attempt to build my baseline classifier, as well as my Logistic Regression model, I enabled all these preprocessing steps – from cleaning, to tokenizing and stop word removal and lemmatization. Surprisingly, these parameters did not yield good results. The best set of parameters that enabled the best performance for my statistical models came from not removing any stop words nor lemmatization at all. The simple neural network however required the data to be more thoroughly cleaned, and did not fare as well with the same set of preprocessing parameters as the traditional models. The best set of preprocessing parameters for my neural network include cleaning, removal of stop words, short words and lemmatization. All evaluations of the ablation studies were performed on my validation set which are untouched from the original "train.csv" file. In this assignment I considered the micro f1-score as the main metric of evaluation used for this classification task. Table 2 shows the breakdown of results for the best set of parameters for each model, as mentioned earlier and the "cleaned" preprocessing parameter means that the augmented training data has been cleaned of any unconflicted texts, special characters new lines and converted to lower case. The different kinds of models used will be explained later in the later section.

Table 1: Setting of best preprocessing parameters for each model

Preprocessing parameters	Naïve bayes micro f1-score	Logistic Regression micro f1-score	Ensemble Classifier Micro f1-score	Simple Neural Network micro f1-score
Cleaned, no stop/short words removal, no lemmatization/ stemming	0.750	0.750	0.776	0.682
Cleaned, stop/ short words removal, lemmatization	0.723	0.723	0.711	0.739

For feature engineering, I opted to utilize Count Vectorizer and Term-Frequency Inverse Document Frequency (TF-IDF) for highlighting the significance of words within the dataset. I also explored using pre-trained word vectors such as GloVe, however due to constraints of computing power, I had to limit my resources to exploring other feature extractors. My approach included employing a range of n-grams that encompassed both unigrams and bigrams. For empirical testing, I varied the range of n-grams used on the dataset and fit it onto my baseline Naïve Bayes classifier. In theory, a higher n-gram range should be able to capture more context, however it produces worse results than before on my validation set, and started to overfit when the range of n-grams increases. Table 1 shows the different empirical results of comparison between different n-gram ranges. Since an increased n-gram will require higher compute power time complexity, I had to settle for something in between (1, 2). The simple neural network model used seemed to be undeterred by the increase range of n-grams as well, only accounting for very minute differences. I settled for the best n-gram range parameter at (1, 2) as it yielded the best f1-score for my 2 traditional machine learning models.

Table 2: Empirical comparison of using different n-gram ranges for feature extraction

n-grams range	Naïve Bayes micro f1-score	Logistic Regression micro f1-score	Simple neural network Micro f1-score
(1, 2)	0.750	0.750	0.739
(1, 3)	0.722	0.722	0.729
(1, 4)	0.712	0.698	0.728
(2, 6)	0.682	0.682	0.723

Model Results

For my machine learning model, all three allowed classifiers were explored to see how well each fared. My baseline **Naïve Bayes** classifier as well as **Logistic Regression** uses the Sk-learn's library, while the **Simple neural network** was a mix of exploration from the lecture notebook or strategic building of sequential layers.

Naïve Bayes Classifier: A multinomial Naïve Bayes classifier was built from the Sk-learn library, where the best set of preprocessing parameters that led to its best micro f1-score is provided in Table 1. Empirical comparison of this model includes analysing its performance on the different combination of preprocessing parameters, different data augmentation techniques, and observing the increase in n-gram range for its feature extraction.

Logistic Regression: Logistic Regression was also built from the Sk-learn library with maximum iterations of 1000 and regularization strength of 2. The best set of preprocessing parameters can be observed from Table 1. I later varied the hyperparameters of the logistic regression. Empirical comparison of this model includes analysing its performance on the different combination of preprocessing parameters, tuning the maximum iterations, different data augmentation techniques and observing the increase in n-gram range for its feature extraction.

Voting Ensemble Classifier: The Ensemble classifier was built from Sk-learn library. The implemented voting was set to 'soft' to account for prediction confidence rather than hard voting which rely on majority votes. In this case it is redundant to do so because I am only using 2 classifiers. Effectively a majority vote would constitute 2/2 which is ineffective in deciding the majority confidence of a prediction. I distributed the weights equally among the logistic regression and naïve bayes classifier. In my empirical comparison, I also tried to tune this parameter to assign more weights to the logistic regression since it yielded the best accuracy, however I later saw that assigning greater weights to LR in an ensemble of only 2 classifiers will pull the metric scores to fit more to just the LR, eroding the purpose of doing an ensemble classifier.

Simple Neural Network: I tried to adapt the simple neural network that was provided from the lecture notebook. The scores from the neural network were disappointing, and the best micro f1-score did not come close to the traditional machine learning models. I have also tried to vary and fine tune the different hyperparameters of the simple neural network such as the learning rate, adding drop out and regularization in my attempts to improve the model's performance. For empirical comparison, I have also tried creating a simple neural network using sequential layers.

In all, my Ensemble voting classifier was by far, the best model that worked out for me. Table 3 shows the best results of all the models used for this assignment. The evaluations were performed on my validation unseen split and were later submitted to the Kaggle leaderboard. The best model was the ensemble classifier. Each submission was considered for its improvement on my validation unseen set, before any submissions, to prevent a repetitive spamming of hopeful submissions.

Table 3: Best results of each model

Evaluation Metric	Naïve Bayes	Logistic Regression	Ensemble Classifier	Simple Neural Network
Accuracy	0.730	0.750	0.772	0.726
Micro f1-score	0.750	0.750	0.772	0.739
Kaggle leaderboard	0.780	0.827	0.843	0.632

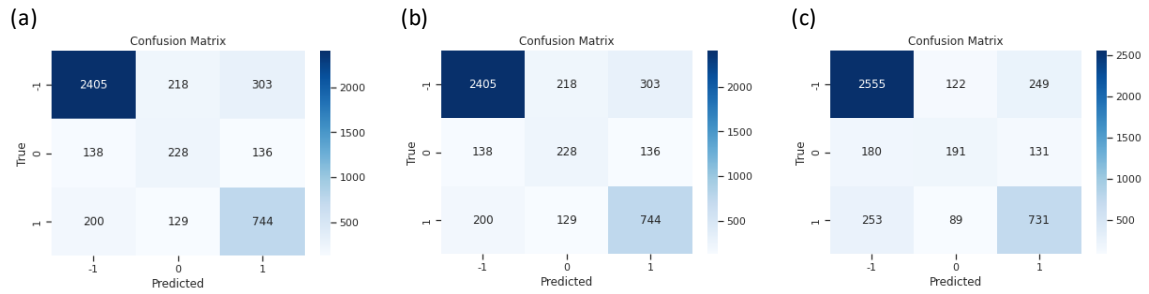


Fig. 4: Confusion matrix of best results of the 3 traditional classifiers (a) Naive Bayes, (b) Logistic Regression, (c) Voting Classifier

Figure 4 shows the confusion matrix of all 3 traditional machine learning classifiers for their best empirical result. It seems that the model is performing worse for class 0, considering the number of correct predictions is almost similar to the number of incorrect predictions for the other two classes. The performance on class 1 is also not ideal, but it is better than for class 0 since there are more true positives relative to the false ones. This is expected because the of the highly unbalanced original set of data shown in Figure 1. As such, class 0 contains the most artificial data followed by class 1. It is only expected for the models to be uncertain in these classes.

Ablation Studies and Analysis

For this section I will elaborate how I fine tune the different parameters for each of my models. The parameters include different steps included or omitted in my preprocessing, features extracted, types of data augmentation techniques, and the hyperparameters of specific models.

Naïve Bayes: I started off with creating a NB vanilla model to see the water level of the NB classifier. From there I tuned the different parameters to create a substantial NB baseline for other models to be compared against. We observe from Table 2 that feature extraction did not really contribute much to increased model performance. Looking at the table below, we can see that preprocessing affects the NB much more significantly

Preprocessing Parameters	n-grams feature extractor range	Data augmentation	Micro f1-score	Remarks
Cleaned, no stop/short words removal, lemmatization	(1, 2)	EDA	0.714	NB Baseline
No preprocessing	No feature extractor	No Data augmentation	0.33	NB Vanilla
Cleaned, no stop/short words removal, no lemmatization/ stemming	(1, 2)	EDA	0.750	NB Best

Logistic Regression: For LR, I tried to tune the specific model hyperparameters as well the different combinations of preprocessing parameters. We can see the specific tuning of the model's hyperparameters did not improve the overall performance of the LR classifier, and preprocessing steps still remains significantly more important.

Preprocessing Parameters	Maximum iterations	Data augmentation	Micro f1-score	Remarks
Cleaned, no stop/short words removal, with lemmatization	3000	EDA	0.714	LR empirical comparison
Remove stop words, lemmatization	2000	EDA	0.687	LR Baseline
Cleaned, no stop/short words removal, no lemmatization/ stemming	1000	EDA	0.750	LR Best

Ensemble Voting Classifier: For VC, the main study for this ablation was to see if the assignment of different weights will improve the overall performance. As theorized earlier, the assignment of higher weights to the better performing LR model will pull the metric scores closer to it. It is essential to consider the number of classifiers in my ensemble voting classifier before deciding the weight distribution of this hyperparameter.

Preprocessing Parameters	Weight assignment [NB, LR]	Data augmentation	Micro f1-score	Remarks
Cleaned, no stop/short words removal, lemmatization	[1, 2]	EDA	0.749	VC empirical comparison
Cleaned, no stop/short words removal, lemmatization	[1, 1]	EDA	0.772	VC Best

Simple Neural Network: For neural network, it was already evident that this model prefers its data to be augmented and cleaned thoroughly. I then experimented with tuning the different hyperparameters. Tuning the hyperparameters did not give me much insight into the direction of improving the performance. I also tried building a sequential layered model which fared worse than the best NN result.

Preprocessing Parameters	Learning Rate	Dropout layers	Number of Epochs	Micro f1-score	Remarks
Cleaned, stop/short words removal, lemmatization	0.001	No	10	0.729	NN Baseline
Cleaned, stop/short words removal, lemmatization	0.0001	No	10	0.730	NN empirical comparison
Cleaned, stop/short words removal, lemmatization	0.0001	Yes	15	0.723	NN empirical comparison
Cleaned, stop/short words removal, lemmatization	0.00001	Yes	10	0.739	NN Best

In conclusion I experimented all 3 different models and realize that retaining the stop words and lemmatization helps best for my traditional models. This might be due to the added sentiment it gives to the overall text, which aids the model to distinguish between the classes. This is evident from Figure 2 as they make up the majority of the training corpus. My ablation study have also provided insights into which component contributes greatly to the performance of each model.

1A. Declaration of Original Work. By entering my Student ID below, I certify that I completed my assignment independently of all others (except where sanctioned during in-class sessions), obeying the class policy outlined in the introductory lecture. In particular, I am allowed to discuss the problems and solutions in this assignment, but have waited at least 30 minutes by doing other activities unrelated to class before attempting to complete or modify my answers as per the Pokémon Go rule.

Signed: A0217019L

I acknowledge that I used the following websites or contacts to complete this assignment (but please note that many uses of Web search and detailed discussion are not allowed:

CS4248 Lecture Notebook 1.1 on Basics of MLP, taking inspiration from the simple neural network model