NLP Project

Detailed Report

Team 20

Name	Section	Bench No.
Essam Wisam	2	2
Mohamed Saad	2	15
Ziad Atef	1	14
Abeer Hussein	2	1

Supervised by

Dr. Sandra Wahid

Eng. Omar Samir

Dec 2022

Table of Contents

Project Pipeline

The following table summarizes the approaches we have tried for each phase of the pipeline

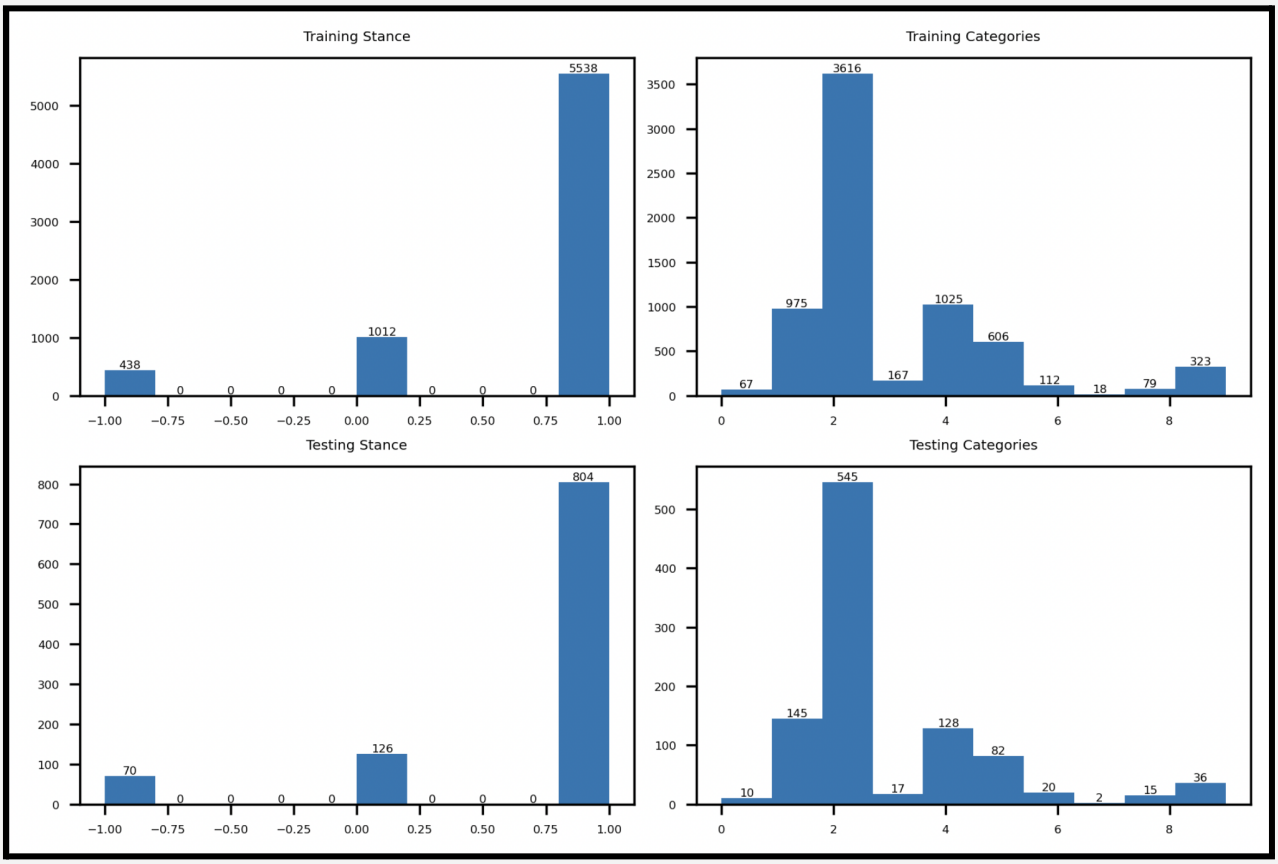
Preprocessing	1. Hand-made 2. NLTK Library 3. SnowBall Library
Feature Extraction	1. BoW 2. TF-IDF [Uni, Bi, Tri, MIX] 3. One-hot Vectors 4. Aravec 5. FastTEXT 6. AraBERT Contextual Embeddings
Model Training	1. Naive Bayes 2. Gradient Boosting 3. SVM 4. GRU & LSTM 5. FFNN 6. AraBERT
Model Evaluation	1. Validation Set

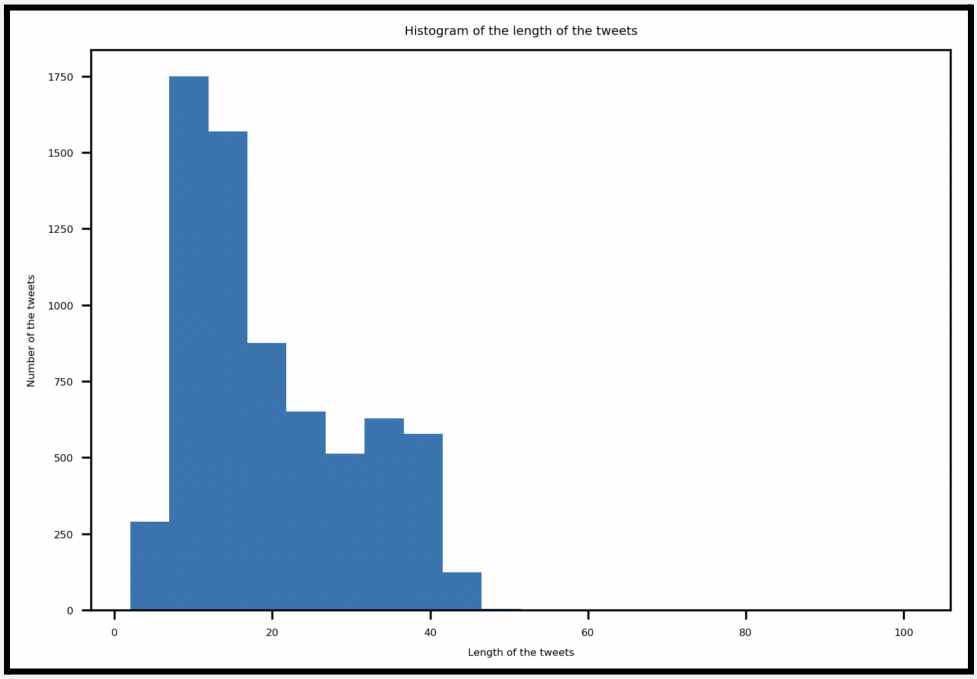
	2. Cross Validation

We will proceed to explain each in the following section in the document.

Pipeline Phases

Before data preprocessing, we started with getting insights on our data that should help us in the future with making decisions. We considered a histogram of classes for each of the two tasks, a histogram for the tweet's length and the most common words for each class. Some of the results are shown below






```
Class 1 of Task 0

[('pad', 76093),
 ('6040 ,'لقاح'),
 ('4393 ,'كورونا'),
 ('1852 ,'علي'),
 ('1198 ,'الجرعه'),
 ('1073 ,'الصحه'),
 ('940 ,'الاولي'),
 ('893 ,'اللقاح'),
 ('883 ,'كوفيد'),
 ('865 ,'يتلقي'),
 ('814 ,'ان'),
 ('730 ,'الله'),
 ('722 ,'الي'),
 ('677 ,'جرعه'),
 ('647 ,'تطعيم'),
 ('620 ,'فيروس'),
 ('566 ,'فايزر'),
 ('553 ,'ضد'),
 ('482 ,'عاجل'),
 ('464 ,'الثانيه')]
```

I. Data Preprocessing

For data preprocessing we have incorporated three modules with the following general pipeline

1. Hand-made Preprocessing

In hand-made preprocessing we used ordinary Regex to clean tweets from URLs, Emojis and such then performed word normalization to compensate for how different words can use different letters for some words (e.g. ة and ه) followed by stop-word removal as provided by the NLTK library since such words would very likely not improve the classification task. We used Penn TreeBank's tokenizer to give the final result.

2. NLTK Preprocessing

The main difference between this and the previous preprocessing method is that we incorporated IRIS stemming by NLTK to reduce the size of the vocabulary. As will be seen later, this method was usually worse than the hand-made method.

3. Snowball Preprocessing

In this one, the main distinction is that we have used a specialized Arabic stemmer from the snowball library. As will be seen later it often has yielded the best results.

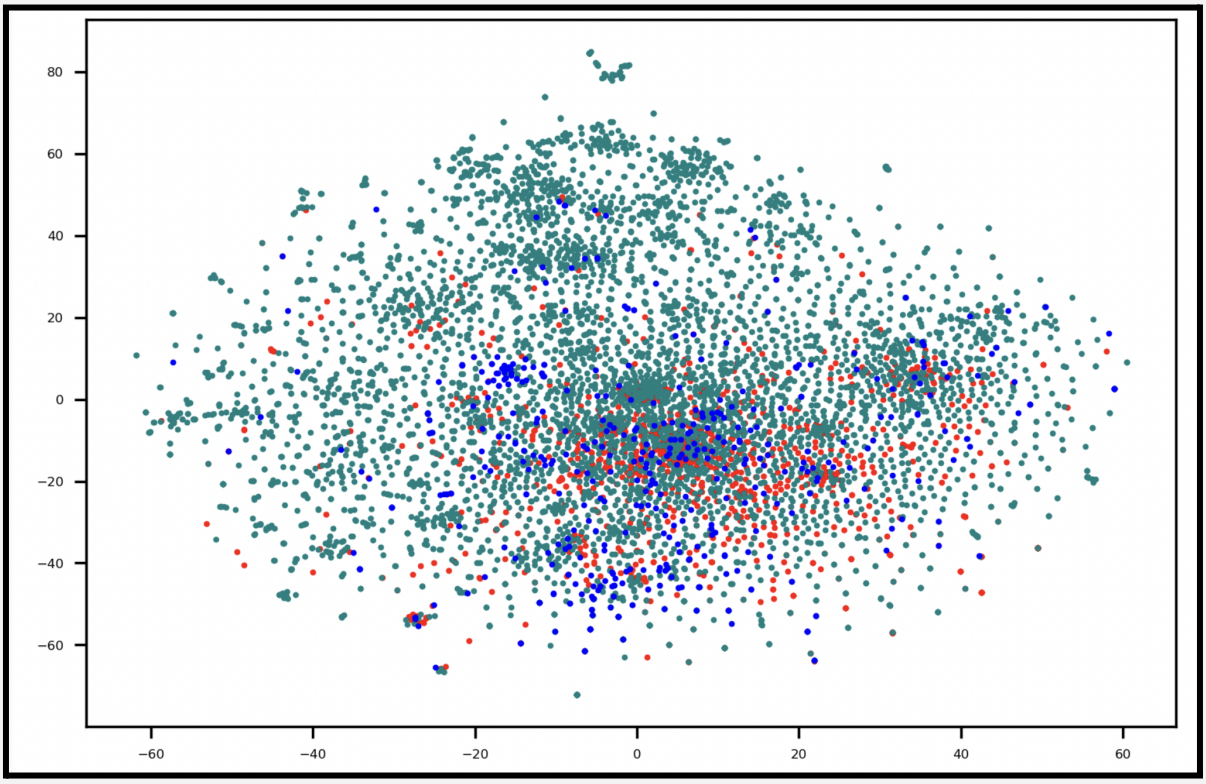
II. Feature Extraction

For features we have considered two different sets of features, one of which would be likely appropriate for classical models and another that focuses more on deep learning models.

1. Bag of Words

In a bag of words, we represented each tokenized tweet with a vector of length as long as the vocabulary (or less) that indicates the no. of times a word in the vocabulary appeared in the tweet. Our main parameters of interest for this feature were min_df and max_df which are the minimum and maximum occurrences of a word to be included in the vocabulary.

The following shows feature visualization results on task 1 after dimensionality reduction with t-SNE



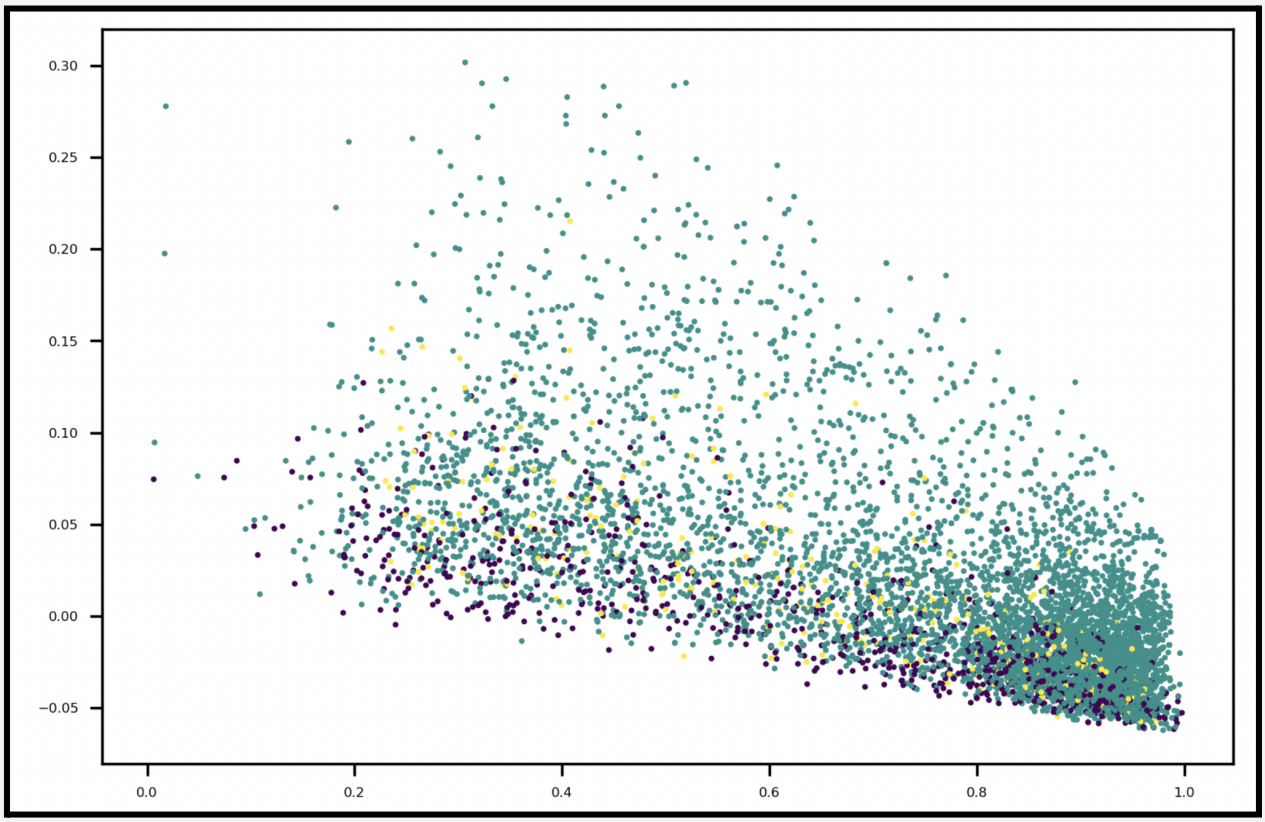
This feature was mainly used with Naive Bayes and SVM

2. TF-IDF

Like BoW, TF-IDF is a classical approach to assign a vector to the whole tweet. For that it relies on the word (generally, n-gram) frequency of each word in the tweet along with the no. of tweets they appear in. To give some sense of normalization, both are multiplied together.

We considered extracting tf-idf features for the unigram, bi-gram and trigram case then proceeded with mixing uni, bi and bi, tri. This was motivated by results we saw in the literature during our literature review prior to starting the project.

The following visualizes the features for task 1 using PCA.



TF-IDF was widely used in all of our classical models (Naive Bayes, SVM and Gradient Descent) and has often shown decent results.

3. BoWTF

A combination of BoW and TF-IDF was used since at some point each were giving decent results. This has improved them slightly as will be seen.

4. One-hot Vectors

This feature represents each word in the preprocessed tweet as a one-hot vector depending on the word's index in the vocabulary. Its primary purpose was to let sequence models such as the GRU or LSTM learn new better embeddings through an embedding layer. We also observed that if each one-hot vector is represented as an integer then we can perceive the sequence vector as being for the whole sentence and used this feature for some classical models as will be shown below.

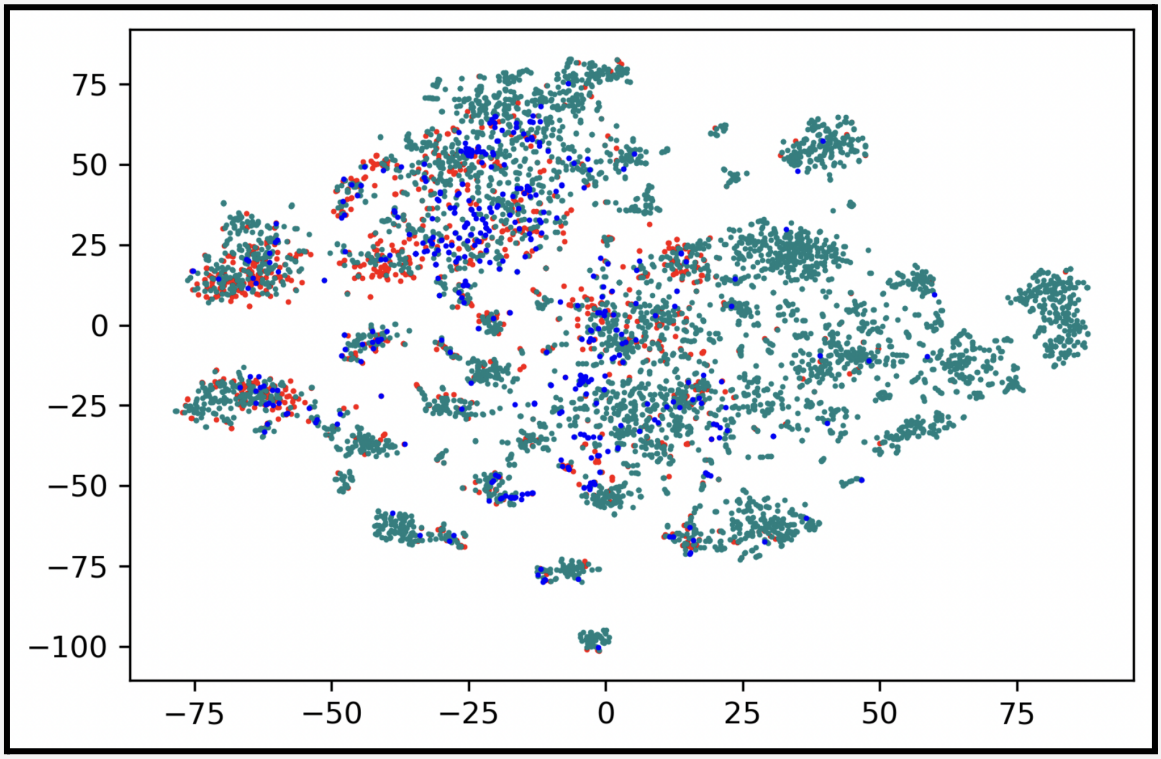
5. Aravec & FastText

Aravec & FastText provide word embedding for Arabic which should be helpful for our sequence models to learn from rather than one-hot vectors or trainable embeddings (since there isn't much data). We generated the word embeddings using the standard setup (as recommended in the documentation).

We quickly switched from Aravec to FastText due to its inability of dealing with unknown words (one of which was كوفيد) but overall neither has added any noticeable improvements compared to trainable embeddings on our sequence models.

6. Contextual Embeddings

Contextual embeddings differ from our previous approach by assigning the vector to the whole sentence (hence, capturing the whole context). We directly used an Arabert that was fine-tuned on a huge corpus of COVID tweets to generate such embeddings and they were used for the feedforward neural network and other models as well. The results after feature visualization on the first task with t-SNE are shown below



III. Model Selection

As illustrated above we used two different sets of models. A classical set composing Naive Bayes, SVM and Gradient Boosting and a deep learning set composing FFNN, LSTM, GRU and ARABERT.

Note as well that for each model we had two versions, one for the first task and another for the second task.

1. Naive Bayes

We used the Naive Bayes Classifier to make predictions for both tasks. Our main hyperparameters of interest were alpha which signifies the amount of smoothing and fit_prior, class_priors which makes it possible to use a uniform prior or custom priors.

We also consider BoW, TF-IDF and a mixture of both for NB.

The best results by NB as shown in the table below used (BoWTF or BoW) and synthetic oversampling via SMOTE and a uniform prior. The last two were essential to deal with the imbalance issue.

Task 1	Task 2
0.59	0.407

2. Gradient Boosting

We used the Gradient Boosting Classifier to make predictions for both tasks. Our main hyperparameters of interest were the number of estimators.

Like all our other classical models we included an option for oversampling using SMOTE or ADASYN prior to passing the data to the model.

For features, our limited trials were restricted to using TF-IDF. We didn't try much more features because it seemed to be slower and wasn't as performant as our other classifiers.

The best results by GB as shown in the table below used TF-IDF and synthetic oversampling via ADASYN for the imbalance issue

Task 1	Task 2
0.5	0.35

3. SVM

We used the SVM Classifier to make predictions for both tasks. Our main hyperparameters of interest were the kernel and the regularization factor. Besides SMOTE for oversampling we also experimented with weight loss since SVM also relies on a loss function.

We tried many features for the SVM classifier (TF-IDF [uni, bi, tri..], BoW, Contextual Embeddings, OneHot) and more details on such trials will be shown below.

The best results by SVM are shown in the table below

Task 1	Task 2
0.55	0.34
Uni TF-IDF, C=1, Poly Kernel & No Oversampling + Weighted Loss	Uni TF-IDF, C=1, Poly Kernel & No Oversampling + Weighted Loss

4. LSTM & GRU

We used the LSTM & GRU as our sequence models to make predictions for both tasks. Our main hyperparameters were the learning rate, number of hidden layers, hidden dimension and drop out probability. To mitigate the imbalance issued there was an option for weighted loss.

The only features we tried for these two were one-hot, Aravec and FastTEXT.

The best results by LSTM & GRU are shown in the table below and were due to the GRU and more details about the specific trial are shown in the evaluation section.

Task 1	Task 2
0.53	0.25

As will be clear in the trials, quite oddly the more we tried to improve these two models the worse were their results. This was expected to some extent since the volume of data we're supplying to these two models is perhaps very small compared to actual use-cases.

5. FFNN

This was a classic feedforward neural network used with the contextual embeddings obtained with AraBERT which makes the whole approach equivalent to fine-tuning the original model (all layers except the new head, i.e., this FFNN are frozen).

We used optional weighted loss to deal with the weighted loss as with the other deep learning models.

Best results are quite embarrassing to report and will be provided in the trials section.

6. AraBERT

This time we used the pre-trained AraBERT fine-tuned on COVID-19 Arabic tweets to make predictions for both tasks. Our approach for the first task relied on using class weights and for the second task it relied on a special training scheme that goes by alternating the model training on the real dataset and a few-shot balanced dataset to counteract the imbalance issue. In both cases, we retrained the whole model.

Our trials for this approach were more limited than others since we had to run the model on the cloud and it took a very long time. However, this has yielded the best results.

Task 1	Task 2
0.46	0.37 [Training Scheme]
0.61 [After Class Weighting]	-

Chosen Model

By the best results from each model we chose AraBERT for task 1 and Naive Bayes for task 2.

Due to issues with generating the csv file for Naive Bayes near the deadline we were only able to submit the results by Arabert for both.

Trials & Evaluation

The following shows large snapshots from the CSV files that track our runs for each model.

1. Naive Bayes

	Approach	Max_len	Method	Model	Alpha	Acc	BF1	WF1
	Hand-made	30	Bag of Words	NB-1	1	0.8	0.3	0.72
	Hand-made	30	Bag of Words	NB-2	1	0.55	0.07	0.38
	Snowball	50	Bag of Words	NB-1	1	0.81	0.44	0.77
	Snowball	50	Bag of Words	NB-1	1	0.81	0.44	0.77
*								

Approach	Max_len	Method	Model	Alpha	Fit Prior	Oversampling	Cross Validati	Acc	BF1	WF1
Snowball	30	Bag of Words	NB-1	1	False	True	False	0.78	0.59	0.79
Snowball	30	Bag of Words	NB-1	1	False	True	False	0.78	0.59	0.79
Snowball	30	Bag of Words	NB-1	1	False	True	True	0.71	0.71	0.71
Snowball	30	Bag of Words	NB-1	0.05	False	True	False	0.78	0.54	0.79
Snowball	30	Bag of Words	NB-1	1	False	True	False	0.78	0.59	0.79
Snowball	30	Bag of Words	NB-2	1	False	True	False	0.54	0.36	0.57
Snowball	30	Bag of Words	NB-2	0.05	False	True	False	0.58	0.41	0.6
Hand-made	30	Bag of Words	NB-1	1	False	True	False	0.75	0.55	0.78
Hand-made	30	Bag of Words	NB-2	0.05	False	True	False	0.59	0.37	0.61
Snowball	30	Bag of Words	NB-1	1	False	True	False	0.78	0.59	0.79
Snowball	30	Bag of Words	NB-2	0.05	False	True	False	0.58	0.41	0.6
Snowball	30	Bag of Words	NB-1	1	Custom	True	False	0.76	0.58	0.78
Snowball	30	Bag of Words	NB-1	1	Custom	True	False	0.74	0.57	0.77
Snowball	30	Bag of Words	NB-1	1	Custom	True	False	0.71	0.55	0.75
Snowball	30	Bag of Words	NB-1	1	False	True	False	0.78	0.59	0.79
Snowball	30	Bag of Words	NB-2	0.05	custom	True	False	0.54	0.36	0.57
Snowball	30	Bag of Words	NB-2	0.05	custom	True	False	0.54	0.36	0.56
Snowball	30	TF-IDF Uni	NB-1	1	False	True	False	0.76	0.58	0.79
Snowball	30	TF-IDF Uni	NB-2	0.05	False	True	False	0.55	0.35	0.57
Snowball	30	Bag of Words	NB-2	0.05	False	True	False	0.58	0.41	0.6
Snowball	30	BoW+TF	NB-1	1	False	True	False	0.78	0.59	0.8
Snowball	30	Bag of Words	NB-2	0.05	False	True	False	0.58	0.41	0.6
Snowball	30	TF-IDF Uni	NB-2	0.05	False	True	False	0.55	0.35	0.57
Snowball	30	Bag of Words	NB-2	0.05	False	True	False	0.58	0.41	0.6

Note the first column signifies the preprocessing approach. BF1 and WF1 are the balanced and weighted Macro F1s respectively. The model is designated with -1 or -2 depending on the task.

2. Gradient Boosting

approach	max_len	method	Model	Number of Boosters	acc	BF1	WF1		
Hand-made	30	One-hot Vector	GB-1	300	0.796	0.339942149347874	0.727510875185976		
approach	max_len	method	Model	Number of Boosters	acc	BF1	WF1		
Hand-made	30	One-hot Vector	GB-2	300	0.617	0.22661331000205237	0.5680244089510303		
approach	max_len	method	n-gram	oversampling	Model	Number of Boosters	acc	BF1	WF1
Snowball	50	TF-IDF	Uni	False	GB-1	300	0.808	0.41530006912172523	0.7553768993214853
approach	max_len	method	n-gram	oversampling	Model	Number of Boosters	acc	BF1	WF1
Snowball	50	TF-IDF	Uni	True	GB-1	300	0.797	0.5020668590228637	0.7754971520750357
approach	max_len	method	n-gram	oversampling	Model	Number of Boosters	acc	BF1	WF1
Snowball	50	TF-IDF	Uni	False	GB-2	300	0.658	0.3404764012755834	0.6284401960839737
approach	max_len	method	n-gram	Model	Number of Boosters	acc	BF1	WF1	
Snowball	30	TF-IDF	Uni	GB-1	200	0.488	0.41110835820335234	0.5506647221127317	
approach	max_len	method	n-gram	Model	Number of Boosters	acc	BF1	WF1	
Snowball	30	TF-IDF	Uni	GB-1	200	0.795	0.49166978749555884	0.7727985480943738	

In this layout, there is a header above each row (because we kept track of more and more things as we developed the model)

3. SVM

approach	max_len	method	Model	Kernel	acc	BF1	WF1	
Hand-made	30	One-hot Vector	SVM-1	poly	0.802	0.326039303	0.72658327	
approach	max_len	method	Model	Kernel	acc	BF1	WF1	
Hand-made	30	One-hot Vector	SVM-2	poly	0.498	0.107365285	0.401961209	
approach	max_len	method	Model	Kernel	acc	BF1	WF1	
Hand-made	30	TF-IDF Uni-gram	SVM-1	poly	0.814	0.403726149	0.752993594	
approach	max_len	method	Model	Kernel	acc	BF1	WF1	
Hand-made	30	TF-IDF Uni-gram	SVM-1	poly	0.814	0.403726149	0.752993594	
approach	max_len	method	Model	Kernel	acc	BF1	WF1	
Hand-made	30	TF-IDF Bi-gram	SVM-1	poly	0.814	0.400429265	0.752624714	
approach	max_len	method	Model	Kernel	acc	BF1	WF1	
Hand-made	30	TF-IDF Tri-gram	SVM-1	poly	0.813	0.392445077	0.750206259	
approach	max_len	method	Model	Kernel	acc	BF1	WF1	
Hand-made	30	TF-IDF n-gram	SVM-1	poly	0.813	0.392157729	0.750516595	
approach	max_len	method	Model	Kernel	acc	BF1	WF1	
Hand-made	30	TF-IDF n-gram	SVM-1	poly	0.813	0.392157729340828	0.7505165948022285	
approach	max_len	method	Model	Kernel	Weighted Loss	acc	BF1	WF1
Snowball	50	TF-IDF n-gram	SVM-1	poly	True	0.81	0.5178450664115455	0.7905426958586126
approach	max_len	method	Model	Kernel	Weighted Loss	acc	BF1	WF1
Snowball	50	TF-IDF n-gram	SVM-2	poly	True	0.537	0.29294114996392356	0.5290071968017752

approach	max_len	method	n-gram	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	acc	BF1	WF1	
Snowball	50	TF-IDF	Uni	SVM-1	rbf	False	False	False	0.812	0.3679524999210923	0.7431351960434408	

approach	max_len	method	n-gram	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	acc	BF1	WF1	
Snowball	50	TF-IDF	Uni	SVM-1	rbf	False	False	False	0.812	0.3679524999210923	0.7431351960434408	

approach	max_len	method	n-gram	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	acc	BF1	WF1	
Snowball	50	TF-IDF	Uni	SVM-1	rbf	True	True	False	0.782	0.5286965062785952	0.7778770558814588	

approach	max_len	method	n-gram	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1
Snowball	30	TF-IDF	Uni	SVM-1	rbf	False	True	False	5	0.806	0.5080989659633938	0.7844681135381444

approach	max_len	method	n-gram	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1
Snowball	30	TF-IDF	Uni	SVM-1	rbf	False	True	False	20	0.81	0.5060802296960185	0.7864265956298871

approach	max_len	method	n-gram	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1
Snowball	30	TF-IDF	Uni	SVM-1	rbf	False	True	False	0.1	0.767	0.4462610017362783	0.7450978645197656

approach	max_len	method	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1	
Snowball	30	Bag of Words	SVM-1	rbf	False	True	False	1.0	0.785	0.4802995311860152	0.7641747357446184	

approach	method	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1	
contextual	contextual	SVM-1	rbf	False	True	False	1.0	0.545	0.07055016181229774	0.3844983818770227	

approach	max_len	method	n-gram	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1
Snowball	30	TF-IDF	Uni	SVM-1	rbf	False	False	False	1.0	0.813	0.3919429796448565	0.7495861089810895

approach	max_len	method	n-gram	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1
Hand-made	30	TF-IDF	Uni	SVM-1	rbf	False	False	False	1.0	0.814	0.4038038038038038	0.7513581081081081

approach	method	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1	
contextual	contextual	SVM-1	rbf	False	True	False	1.0	0.545	0.07055016181229774	0.3844983818770227	

approach	max_len	method	n-gram	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1
Snowball	30	TF-IDF	Uni	SVM-1	rbf	False	False	False	1.0	0.813	0.3919429796448565	0.7495861089810895

approach	method	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1	
contextual	contextual	SVM-1	rbf	False	True	False	1.0	0.545	0.07055016181229774	0.3844983818770227	

approach	max_len	method	n-gram	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1
Snowball	30	TF-IDF	Uni	SVM-1	rbf	False	True	False	1.0	0.806	0.5145374077672679	0.7825990523552493

approach	max_len	method	n-gram	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1
Snowball	30	TF-IDF	Uni	SVM-1	poly	False	True	False	3.0	0.782	0.5327926059438862	0.7813413100911087

approach	method	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1	
contextual	contextual	SVM-1	poly	False	True	False	3	0.036	0.006949806949806949	0.0025019305019305014	

approach	max_len	method	n-gram	Model	Kernel	Weighted Loss	Oversampling	CrossValidation	C	acc	BF1	WF1
Snowball	30	TF-IDF	Uni	SVM-1	poly	False	True	False	3	0.561	0.28875306092218667	0.5686662341566239

4. GRU

approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1			
Hand-made	30	One-hot Vector	GRU-1	128	2	200	256	0.5	0.001	5	5	0.781	0.5304863461440659	0.7780260407466394			

approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1			
Hand-made	30	One-hot Vector	GRU-2	128	2	200	256	0.5	0.001	5	5	0.662	0.2560178176995256	0.6232475229154204			

approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1			
Snowball	30	One-hot Vector	GRU-2	512	2	200	256	0.5	0.001	5	5	0.628	0.1607788437977117	0.5280249184777487			

approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1			
Snowball	30	One-hot Vector	GRU-1	128	2	200	256	0.5	0.001	5	5	0.811	0.4992972512798983	0.7839730888989959			

approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1			
Snowball	50	One-hot Vector	GRU-1	128	2	200	256	0.5	0.001	5	5	0.806	0.4049572307167286	0.749509706166522			

approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	Weighted Loss	acc	BF1	WF1		
Snowball	50	One-hot Vector	GRU-2	512	2	200	256	0.5	0.001	5	5	True	0.009	0.005830013230825427	0.013630902324113075		

approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	Weighted Loss	acc	BF1	WF1		
Snowball	50	One-hot Vector	GRU-1	128	2	200	256	0.5	0.001	5	5	True	0.607	0.45795793833699981	0.6650810702594867		

approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	Weighted Loss	acc	BF1	WF1		
Snowball	50	One-hot Vector	GRU-2	512	2	200	256	0.5	0.001	5	5	True	0.011	0.012510789761941513	0.016963980174984916		

approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	Weighted Loss	acc	BF1	WF1		
Snowball	50	One-hot Vector	GRU-2	512	2	200	256	0.5	0.001	5	5	True	0.121	0.05975601395749639	0.09754370297926 63		

approach	max_len	method	FastText LR	emb_dim	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	Weighted Loss	acc	BF1	WF1
Snowball	30	FastText	0.05	300	GRU-1	128	2	300	256	0.5	0.001	5	5	True	0.556	0.3497486192576185	0.6066421277991557

approach	max_len	method	FastText LR	emb_dim	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	Weighted Loss	acc	BF1	WF1
Snowball	30	FastText	0.05	300	GRU-2	512	2	300	256	0.5	0.001	5	5	True	0.193	0.10652870017424523	0.13078236941881655

5. LSTM

approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1				
Hand-made	30	One-hot Vector	LSTM-1	128	2	200	256	0.5	0.001	5	5	0.804	0.4909361765843308	0.7839143490125163				
approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1				
Hand-made	30	One-hot Vector	LSTM-2	128	2	200	256	0.5	0.001	5	5	0.67	0.2003826600080954	0.5935169535027927				
approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1				
Hand-made	30	One-hot Vector	LSTM-1	128	3	200	256	0.6	0.002	5	5	0.791	0.4261255018423802 4	0.7664765286256393				
approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1				
Hand-made	40	One-hot Vector	LSTM-1	128	2	200	256	0.5	0.001	5	5	0.788	0.3785705006520141	0.7427932069948774				
approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1				
Hand-made	30	One-hot Vector	LSTM-1	128	2	200	256	0.5	0.001	5	5	0.793	0.4362423827141842	0.7661224226680901				
approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1				
NLTK	30	One-hot Vector	LSTM-1	128	2	200	256	0.5	0.001	5	5	0.795	0.3891780482721733	0.7504674908520045				
approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1				
Hand-made	30	One-hot Vector	LSTM-1	256	2	200	300	0.5	0.001	5	5	0.798	0.3426294152796788	0.7314157802278337				
approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1				
Snowball	30	One-hot Vector	LSTM-2	128	2	200	256	0.5	0.001	5	5	0.628	0.1997669752527655	0.5080771653064271				
approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1				
Snowball	30	One-hot Vector	LSTM-1	256	2	200	300	0.5	0.001	5	5	0.803	0.3071869992175765	0.7285440646691854				
approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	acc	BF1	WF1				
Snowball	50	One-hot Vector	LSTM-2	128	2	200	256	0.5	0.001	5	1	0.545	0.0705501618129774	0.3844983818770227				
approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	Weighted Loss	acc	BF1	WF1			
Snowball	50	One-hot Vector	LSTM-1	256	2	200	300	0.5	0.001	5	5	True	0.164	0.15610756606763553	0.14066472054082955			
approach	max_len	method	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	Weighted Loss	Weights	acc	BF1	WF1		
Snowball	50	One-hot Vector	LSTM-1	256	2	200	300	0.5	0.001	5	5	True	tensor([0.3000, 0.2500, 0.7000])	0.804	0.2971175166297117 5	0.7166474501808647		
approach	max_len	method	FastText LR	emb_dim	Model	batch_size	Number of Layers	Embedding Dimension	Hidden Dimension	Dropout Prob	Learning Rate	Gradient Clip	Number of Epochs	Weighted Loss	Weights	acc	BF1	WF1
Snowball	50	FastText	0.05	300	LSTM-1	256	2	300	300	0.5	0.001	5	5	True	tensor([0.3000, 0.2500, 0.7000])	0.804	0.2971175166297117 5	0.7166474501808647

6. FFNN

approach	Model	batch_size	Hidden Dimension 1	Hidden Dimension 2	Learning Rate	Number of Epochs	Weighted Loss	Weights	acc	BF1	WF1	
contextual	Arabert-1	128	768	512	3e-06	300	False	tensor([0.3000, 0.2500, 0.7000])	0.435	0.27235142618773095	0.4937965312571588	
approach	Model	batch_size	Hidden Dimension 1	Hidden Dimension 2	Learning Rate	Number of Epochs	Weighted Loss	Weights	acc	BF1	WF1	
contextual	Arabert-2	128	768	512	3e-06	300	False	tensor([0.3000, 0.2500, 0.7000])	0.132	0.0660753515234169	0.14543352645637597	
approach	Model	batch_size	Hidden Dimension 1	Hidden Dimension 2	Learning Rate	Number of Epochs	Weighted Loss	Weights	acc	BF1	WF1	
contextual	Arabert-1	512	768	512	3e-06	300	True	tensor([0.3000, 0.2500, 0.7000])	0.531	0.3179848093760372	0.5830804033217003	
approach	method	Model	batch_size	Hidden Dimension 1	Hidden Dimension 2	Learning Rate	Number of Epochs	Weighted Loss	Weights	acc	BF1	WF1
contextual	contextual	Arabert-1	512	768	300	3e-06	300	True	tensor([0.3000, 0.2500, 0.7000])	0.17	0.16182975261856664	0.19034226742557175
approach	method	Model	batch_size	Hidden Dimension 1	Hidden Dimension 2	Learning Rate	Number of Epochs	Weighted Loss	Weights	acc	BF1	WF1
contextual	contextual	Arabert-1	512	768	300	3e-06	300	True	tensor([0.3000, 0.7000, 0.7000])	0.311	0.2367505338952784 8	0.3858496903102758
approach	method	Model	batch_size	Hidden Dimension 1	Hidden Dimension 2	Learning Rate	Number of Epochs	Weighted Loss	Weights	acc	BF1	WF1
contextual	contextual	Arabert-1	512	768	300	3e-05	300	False	tensor([0.3000, 0.7000, 0.7000])	0.133	0.13470020323932258	0.13070754699628442
approach	method	Model	batch_size	Hidden Dimension 1	Hidden Dimension 2	Learning Rate	Number of Epochs	Weighted Loss	Weights	acc	BF1	WF1
contextual	contextual	Arabert-1	2048	768	300	3e-06	300	False	tensor([0.3000, 0.7000, 0.7000])	0.723	0.3240041396356631	0.6915544273373817
approach	method	Model	batch_size	Hidden Dimension 1	Hidden Dimension 2	Learning Rate	Number of Epochs	Weighted Loss	Weights	acc	BF1	WF1
contextual	contextual	Arabert-1	2048	768	300	3e-06	500	False	tensor([0.3000, 0.7000, 0.7000])	0.253	0.17415007215007217	0.2732967272727273
approach	method	Model	batch_size	Hidden Dimension 1	Hidden Dimension 2	Learning Rate	Number of Epochs	Weighted Loss	Weights	acc	BF1	WF1
contextual	contextual	Arabert-1	2048	768	1000	3e-06	300	False	tensor([0.3000, 0.7000, 0.7000])	0.54	0.2993163133543553	0.5835312880753553

Thanks.