

Long-Short-Term-Memory Models in Propaganda Detection

1. Introduction

Long-Short-Term-Memory models (LSTMs) has received constant research attention over the years in Natural Language Processing (NLP) (Ma, 2016) (Byeon, 2015) (Alzaidy, 2019). LSTMs are used in NLP tasks such as text classification (Zhou, 2015), Part-of-Speech (POS) tagging (Balwant, 2019, July), and Named-Entity-Recognition (NER) tasks (Limsopatham, 2016, December). LSTMs incorporate additional gates for the purposes of improving the locality issue and vanishing gradient issue in Recurrent Neural Network models (RNNs) (Sherstinsky, 2020). Not only do the incorporated gates offer LSTMs good prediction in sequential input data, such as word sequences, but the Bidirectional LSTMs (bi-LSTMs) enable additional training by traversing input data twice (Huang, 2015) (Mahadevaswamy, 2023). In this study, we examine the application of LSTMs in completing the propaganda detection task.

There are two main tasks in propaganda detection. The first task is classifying a set of word sequences by the propaganda technique each of them uses (defined as Task 1 in this study). We compare two methods in solving this problem: one using a Logistic Regression model, i.e. a linear statistical model, and another using a bidirectional-Long-Short-Term-Memory (bi-LSTM) model, i.e. a neural network model. We also include a Multinomial Naïve Bayes (MultinomialNB) model as a baseline of the performance. Generally, a discriminative model like Logistic Regression model should outperform a generative model like Naïve Bayes model (Ng, 2001) (Xue, 2008). Furthermore, Naïve Bayes models often produce widely overconfident probability due to the naïve independence assumptions (Rish, 2001). The second approach adopts bi-LSTM, a neural network model that specialises in analysing relations within sequential data. In practice, a neural network model can have low performance if the data is not large enough or not complex enough (Boulesteix, 2014) (Du, 2013); still, with the bi-LSTM model we aim to have a better performance than the MultinomialNB model at best.

The second task is using neural network approaches in propaganda detection (defined as Task 2 in this study). The detection pipeline includes two parts: first, identifying a propaganda snippet in a sample sentence, and second, classifying the propaganda technique the sample uses. We present SOBIE labelling system we use to complete this pipeline. We compare two models in using the SOBIE labelling system: an LSTM model and a Maximum-entropy Markov Model bi-gram neural language model (MEMM bi-gram NLM). We also discuss how SOBIE labelling system can fix the sparsity issue encountered when tagging word sequences (Allison, 2006) (Chen, 2024).

2. Methods

2.1. About Data

A dataset of 3200 samples is being provided. Each sample includes a sentence that contains a propaganda snippet and a label. The label suggests which propaganda technique the propaganda snippet uses. If the snippet doesn't use a propaganda technique, it is labelled as "not_propaganda". Excluding the "not_propaganda", there are 8 propaganda techniques with each being used in 200 samples; and there are 1600

samples that have the label “not_propaganda” (Figure 1). In other words, if excluding samples labelled as “not-propaganda”, the class distribution is even. Each sentence in a sample has a <BOS> at the beginning of the propaganda snippet and an <EOS> tag at the end of the propaganda snippet. If the snippet does not use any propaganda technique, it will be labelled accordingly with the <BOS> and <EOS> tags kept. The average sample length for each class spans from 21 to 33 words, whereas the average snippet length varies from 2 to 17 words (Table 1). The longest sample and longest snippet are found in class “doubt”, which the max sample length is 140 words, and the max snippet length is 140 words.

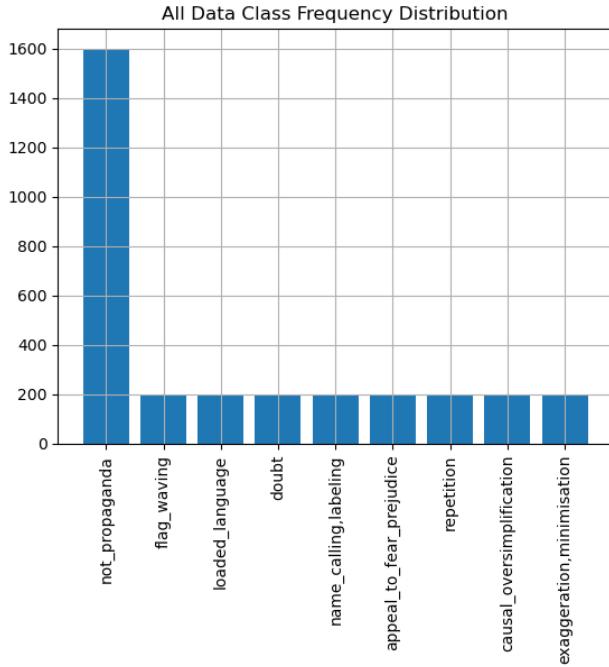


Figure 1. Class distribution of all data set. Including “not_propaganda”, there are 9 classes in total. There are 1600 samples that have label “not_propaganda”, and there are 200 samples in each of the rest of the classes.

(a)	not_propaganda	flag_waving	loaded_language	doubt	name_calling,labeling	appeal_to_fear_prejudice	repetition	causal_oversimplification	exaggeration,minimisation
Average	21.290625	29.995	28.26	31.42	33.745	29.695	24.845	33.56	30.22
Max	113.000000	102.000	81.00	140.00	116.000	100.000	110.000	120.00	78.00
Sum	34065.000000	5999.000	5652.00	6284.00	6749.000	5939.000	4969.000	6712.00	6044.00

(b)	not_propaganda	flag_waving	loaded_language	doubt	name_calling,labeling	appeal_to_fear_prejudice	repetition	causal_oversimplification	exaggeration,minimisation
Average	6.53875	11.22	3.545	21.135	4.635	17.935	2.95	22.115	7.825
Max	83.00000	73.00	28.000	140.000	25.000	74.000	32.00	72.000	39.000
Sum	10462.000000	2244.00	709.000	4227.000	927.000	3587.000	590.00	4423.000	1565.000

Table 1. Tables show the average length and maximum length of (a) the sentences and (b) the propaganda snippets within sentences in the samples.

Before we feed data to any model, we preprocess both the text and the labels. Firstly, we keep only the samples that have propaganda (i.e. samples not being labelled as not_propaganda). There are two reasons for this removal. Firstly, there are far more non-propaganda samples (1600 samples) than any other classes (200 samples in each of the classes). Such heavy tailed class distribution may lead to poor training (Murphy, 2004). Secondly, we are not solving a binary classification problem of labelling data as either “is_propaganda” or “not_ propaganda”; instead, we are solving a multi-class classification problem of identifying the correct propaganda technique a span of text is

using, and the correct part of the sentence that should be the propaganda snippet. Therefore, it is plausible that we train the models without non-propaganda samples. After the removal, we process the text by lowercasing the text, splitting each sentence by words, and removing special characters and punctuations, including the <BOS> and <EOS> tags. The propaganda snippets will be extracted using SOBIE labelling system, which we will explain in Section 2.3.1. Thirdly, we assign each of the propaganda technique with a numeric label. Table 2 shows a list of numeric labels and each corresponding propaganda technique.

Propaganda Technique Class Name	flag_waving	loaded_language	doubt	name_calling,labeling	appeal_to_fear_prejudice	repetition	causal_oversimplification	exaggeration,minimisation	
Assigned Label	0	1	2	3	4	5	6	7	

Table 2. The table shows the assigned labels for each propaganda technique. After removing non-propaganda samples, there are 8 classes in total. Each class is assigned with a unique numeric label between 0 to 7. The numeric label also matches the index of the output tensor of the classifiers used in Task 1.

For Task 1, all sequence classifiers are being trained using both labelled propaganda sentences and labelled propaganda snippets (and both texts being pre-processed). For example, a sample of a processed sample text of "the obama administration misled the american people and congress because they were desperate to get a deal with iran said sen" has a processed snippet "american people" with propaganda technique label = 0 ("flag-waving"). The model will be trained on this sample twice with the sentence and snippet and both being labelled as 0. For Task 2, the sequence taggers are being trained using labelled propaganda sentences only.

The final dataset has 3200 samples after the non-propaganda removal and the augmentation of sentence samples and snippet samples. The data is being split by 4:1 for training and testing. The former contains 2560 samples, and the latter contains 640 samples. For training the neural models in Task 1, we split the testing dataset in half for validation and evaluation (i.e. testing). The training dataset and validation dataset are each being sliced into batches with batch size of 32.

2.2. Task 1. Propaganda Technique Classification

In this section, we describe the two main approaches (i.e. defined as Method 1 and Method 2 in Section 2.2.) in propaganda technique classification on a word sequence, i.e. a sentence that contains a propaganda snippet. We evaluate all models using their F1 scores on each class (i.e. each propaganda technique) and the macro average F1 scores. In Method 1, we use a Logistic Regression model and a MultinomialNB model and we compare them with their mean square errors on testing data. In Method 2, we have two variations of bi-LSTM model, differentiated by the way we encode words. When training the model using different encoding method, we observe the cross-entropy loss on training data and validation data throughout the training.

2.2.1 Method 1. Statistical Model Approach

In this section, we show how we use simple statistical models to solve the multi-class sequence classification problem, including a MultinomialNB classifier, a generative model, and Logistic Regression classifier, a discriminative model. We create a bag-of-words representation of each sentence using CountVectorizer (Goyal, 2021, October) (Patel, 2021, June). For example, a text of "the plague is impossible to eradicate" would be represented as a 2 dimensional vector (0, 2168), (0, 3133), (0, 3369), (0, 4614), (0,

6170), (0, 6259). Each (x, y) can be plot on an x-y plane that represents a word in the sentence. The x value represents which sample the word is in, and the y value represents the frequency count of that word being used in whole dataset (i.e. training data and testing data).

2.2.2 Method 2. Neural Network Model Approach

The second approach for solving Task 1 is using a neural language model, a three-layered bi-LSTM. Particularly, we have three LSTM layers, with each layer consisting of a forward direction LSTM and a backward direction LSTM. Each hidden layer in the LSTMs has a dimension of 128. The model takes a word sequence, i.e. a sentence, in and creates embeddings for the input with an embedding dimension of 128. The model has a fully connected linear layer at the output with a dropout of 0.5. The learning rate is 0.001 and we use an Adam optimiser for optimisation. The loss function we adopt is Cross Entropy Loss Function.

We use a different way of word encoding than Method 1. Instead of using CountVectorizer to create a bag-of-words representation for each sample text, we assign a unique value as a representation for each bag-of-words representation. The former encoding treats the same vocabulary in different samples differently, whereas the latter treats them equally. In other words, the latter focus mainly on the relationships between words in a sentential sequence rather than their spatial relationship in different sentences.

To avoid overfitting, we use an early stopping technique with the validation data during the training session (Yao, 2007) (Prechelt, 2002). The early stopping validates every 5 epochs. The hyperparameters for early stopping are set as following: patience = 4 epochs and best validation loss is initialised as infinity in type Float.

2.3. Task 2. Propaganda Snippet Identification and Technique Classification

There are two parts in this section. In the first part, we present SOBIE labelling system that is used to identify the propaganda snippet in a sentence and the propaganda technique the snippet uses. We also discuss how we can encounter a sparsity issue when using this labelling system during training and how we improve the labelling system to fix the issue. In the second part, we present two models that uses SOBIE labelling system in sequence labelling task: a single-layered LSTM model (defined as Method 1 in Section 2.3.) and an MEMM bigram NLM (defined as Method 2 in Section 2.3.). Both models have a demo function that takes a sentence as an input and returns its prediction of the propaganda snippet within the sentence and the technique the snippet uses (see Code Appendix).

We evaluate the performance with the models' F1 scores of, first, each of the 5 SOBIE label classes (i.e. 'S', 'O', 'B', 'I', and 'E') and second, each of the 8 propaganda classes (i.e. propaganda technique). Particularly, regarding the F1 scores of the SOBIE label classes, we examine the F1 scores of 'B' and 'I' class as these are the tags that suggest the propaganda snippet. Since the performance in tagging the correct 'S', 'E', and 'O' are not the focus of this task; we therefore compare the models with macro average F1 scores that are averaged from the F1 scores of class 'B' and 'I'.

2.3.1. The SOBIE Labelling System

In this section, we explain the SOBIE labelling system used for propaganda snippet identification. Similar to BOI tagging for NER task (Alshammari, 2021) (Sun, 2022), SOBIE stands for “Start”, “Outside”, “Beginning”, “Inside”, and “End”. The “Start” label marks the beginning of a sentence, and the “End” label marks the end of a sentence. A word is tagged with an “Outside” label if it’s not part of a propaganda snippet. A word is tagged with a “Beginning” label if it’s the first word of a propaganda snippet, or an “Inside” label if it’s not the first word but part of a propaganda snippet. For example, we have a tagged sentence as follow:

“The Obama administration misled the <BOS> American people <EOS> and Congress because they were desperate to get a deal with Iran,” said Sen.”

After text preprocessing, including lowercasing the text, removing <BOS> tags, <EOS> tags, punctuation, and splitting the sentence by words, we should get a word sequence and a snippet as follow:

*Word sequence: ” 'the', 'obama', 'administration', 'misled', 'the', 'american', 'people', 'and', 'congress', 'because', 'they', 'were', 'desperate', 'to', 'get', 'a', 'deal', 'with', 'iran', 'said', 'sen'"
Snippet: “american”, ‘people’”*

The word sequence should be tagged as follow:

Tagged word sequence: ””-‘S’, 'the'-‘O’, 'obama'-‘O’, 'administration'-‘O’, 'misled'-‘O’, 'the'-‘O’, 'american'-‘B’, 'people'-‘I’, 'and'-‘O’, 'Congress'-‘O’, 'because'-‘O’, 'they'-‘O’, 'were'-‘O’, 'desperate'-‘O’, 'to'-‘O’, 'get'-‘O’, 'a'-‘O’, 'deal'-‘O’, 'with'-‘O’, 'iran'-‘O’, 'said'-‘O’, 'sen'-‘O’, ” -‘E’”

Hence, given the word sequence as an input, we expect our sequence labelling model to predict a sequence of label:

Label Sequence: “S', 'O', 'O', 'O', 'O', 'O', 'B', 'I', 'O', 'E”

However, in Section 2.1, as shown in Table 1, we have found that average snippet length is far shorter than average sentence length. This means there will be far more words tagged as ‘O’ than words tagged as ‘B’ and ‘I’. This sparsity issue can impede the model to learn the most important information, i.e. the words that are tagged as ‘B’ and ‘I’ that belong to propaganda snippets. Therefore, we include the propaganda technique label into the labelling system. For example, the above sentence uses propaganda technique “flag_waving”. According to Table 2, the numeric label for this class is 0. Therefore, the new label sequence would be as follow:

Label Sequence: “0S', '0O', '0O', '0O', '0O', '0O', '0B', '0I', '0O', '0E”

Since we have 5 tags (i.e. ‘S’, ‘O’, ‘B’, ‘I’, and ‘E’) and 8 classes (i.e. 8 propaganda techniques), there are 40 labels in the SOBIE system. During the evaluation of snippet identification, we do not take the classes into account; in other words, ‘0B’ and ‘1B’ would be treated as equally as ‘B’. The improved labelling system is designed to encourage the model to learn the important labels, the ‘B’s and the ‘I’s, during the training.

Furthermore, expanding the labels from 5 to 40 also supports the model to learn the propaganda technique each sentence uses. During the evaluation of propaganda technique classification, we do not take the sequence into account; in other words, ‘1S’, ‘1O’, ‘1B’, ‘1I’, and ‘1E’ will be treated equally as ‘1’. The sequence is classified to the most frequently tagged class label. For example, if a word sequence is labelled as ‘1S’, ‘2O’, ‘2O’, ‘2O’, ‘3O’, ‘2B’, ‘7O’, ‘2O’, ‘2E’, the sequence is classified as class label ‘2’.

2.3.2. Method 1. The LSTM Sequence Tagger

This section we explain how an LSTM model uses the SOBIE system in sequence tagging. The model takes a sentence as an input, whereas the input word sequence is encoded with a bag-of-words representation. The bag-of-words is created by going through the whole dataset (i.e. 1600 samples) and each vocabulary is assigned with a unique value. The model outputs a label sequence with each element picked from one of the 40 labels in SOBIE system. Each SOBIE label is assigned with a unique value (i.e. one from 0 - 39). An LSTM is not required to have a fixed length input and output (Ergen, 2017) (You, 2019, November); as a result, the LSTM Tagger has a dynamic input and output size in accordance with the sentence length it takes in. The avoidance of padding, the input can also prevent the model from having noises in learning label tagging (Dwarampudi, 2019).

The LSTM tagger starts with a single layered of LSTM layer with a hidden layer dimension of 15 and a linear layer on the next. The input is translated into word embedding with an embedding dimension of 15. A softmax layer is capped at the output so the output is a matrix of probability in size of input size (i.e. sentence length) with a dimension of 40 (i.e. number of SOBIE labels). We use Negative Log Likelihood Loss Function as the model loss function and Adam optimiser for optimisation. Learning rate is set to 0.001 and the model is trained for 100 epochs.

2.3.3. Method 2. The MEMM Bi-gram Neural Language Model

This section describes how an MEMM bi-gram NLM uses the SOBIE label system for completing Task 2. MEMMs calculate the conditional probability of a label given the current observation (i.e. word) and its context (previous labels) (McCallum, 2000, June). Our model is called an MEMM bigram model as it simplifies the conditional probability calculation; instead of using a full sequence of labels as context, we use only the previous label. In short, we train the neural network to find the probability of $P(\text{current_label} | \text{previous_label}, \text{current_word})$. As the probability can be small, we take the log value of the probability as the output. The neural network consists of two fully connected linear layers with a ReLU function in between. Each layer has a dimension of 128. The model takes an input of a word representation and a label index and translates the word representation to a word embedding in an embedding dimension of 15. Then, the model creates a concatenation tensor of the word embedding and the label index and passes it to the first hidden layer. Similar to Method 1, we use Negative Log Likelihood Loss Function as the model loss function and Adam optimiser for optimisation. The learning rate is 0.001 and the model is trained for 100 epochs.

3. Results and Analysis

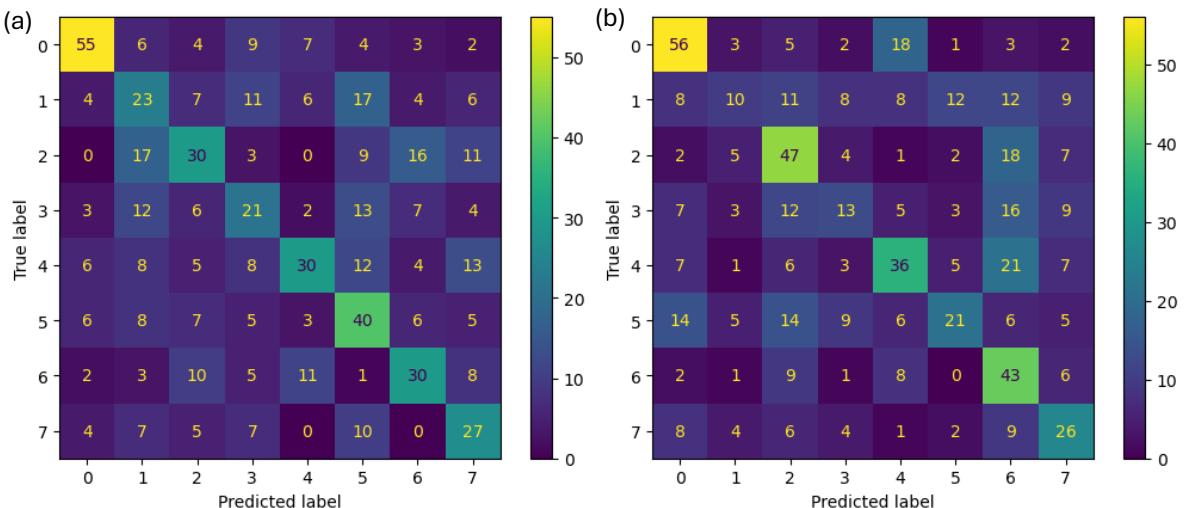
3.1. Task 1. Propaganda Technique Classification

The results in Figure 2 and Table 3 aligns with our assumption that a Logistic Regression model performs better than a MultinomialNB model. The Logistic Regression classifier has an average F1 score of 0.410 whereas the MultinomialNB classifier has an average F1 score of 0.383. Out of 8 classes, there are 5 classes that Logistic Regression classifier has a higher F1 score. Logistic Regression classifier also has a lower mean square error ($\text{mse} = 6.429$) than that of the MultinomialNB classifier ($\text{mse} = 7.089$). Overall, true positive for each class are not zero, suggesting that both classifiers can recognise each of the classes and distinguish the data between classes. Furthermore, class “flag-waving” (label = 0) has the best F1 scores in both models’ results.

While MultinomialNB provides the bi-LSTM model a baseline performance (average F1 score = 0.383), the neural network model, however, did not pass the baseline performance (average F1 score = 0.339). Bi-LSTM model have slightly better performance in class “name_calling” (label = 3) and class “causal_oversimplification” (label = 6); other than that, the rest of F1 scores are less than that of MultinomialNB model.

Propaganda Technique (Assigned label)	0	1	2	3	4	5	6	7	Average
Logistic Regression	0.647	0.284	0.375	0.307	0.414	0.410	0.429	0.398	0.410
MultinomialNB	0.578	0.182	0.480	0.232	0.426	0.333	0.434	0.370	0.383
bi-LSTM	0.437	0.245	0.418	0.353	0.235	0.295	0.448	0.308	0.339

Table 3. This table shows a comparison of macro average F1 score and F1 scores for each class performed by each model in Task 1. Logistic Regression Model has the highest average F1 score. The reason of the neural network model having the worst performance can further be investigated. It is likely that the data size is too small, or the data is not complex enough for the neural network model to achieve a good performance.



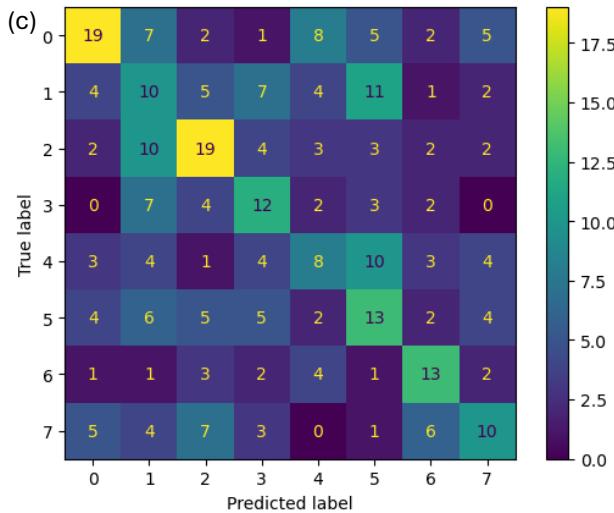


Figure 2. The Figures show the Task 1 results in 3 confusion matrices, and respectively they are from (a) the Logistic Regression model, (b) the MultinomialNB model, and (c) the bi-LSTM model. The numbers from 0-7 represents each of the propaganda technique classes (see Table 2).

To sum up, the results show that neural network approaches does not guarantee a better performance than other machine learning algorithms. Though a neural network classifier may have sufficient capacity to understand the data, the dataset might not be large enough or the data points are not complex enough to be indescribable in vector metrics (Olson, 2018). However, when data points cannot be explained on a distance metric and the dimensions are large, only neural network approaches can solve the problem (Pasini, 2015). Sequence labelling involves mapping a word sequence to a label sequence, and it requires the model to capture relationships in high dimensional spaces (i.e. word embeddings), which we will demonstrate in Task 2.

3-2. Task 2. Propaganda Snippet Identification and Technique Classification

Figure 3-a shows the results of the LSTM tagger's propaganda snippet prediction. The numbers from 0 to 4 stands for, respectively, SOBIE labelling system's 'S', 'O', 'B', 'I', 'E'. We can see that the F1 scores for class 'B' and class 'I' are 0.165 and 0.443. The performance in detecting the first word of a propaganda snippet is not as good as detecting the rest of the snippet. For the MEMM bi-gram NLM, it has F1 scores 0.153 and 0.722 for class 'B' and class 'I', showing a better performance in class 'I' and almost equal performance in class 'B' (Figure 3-b).

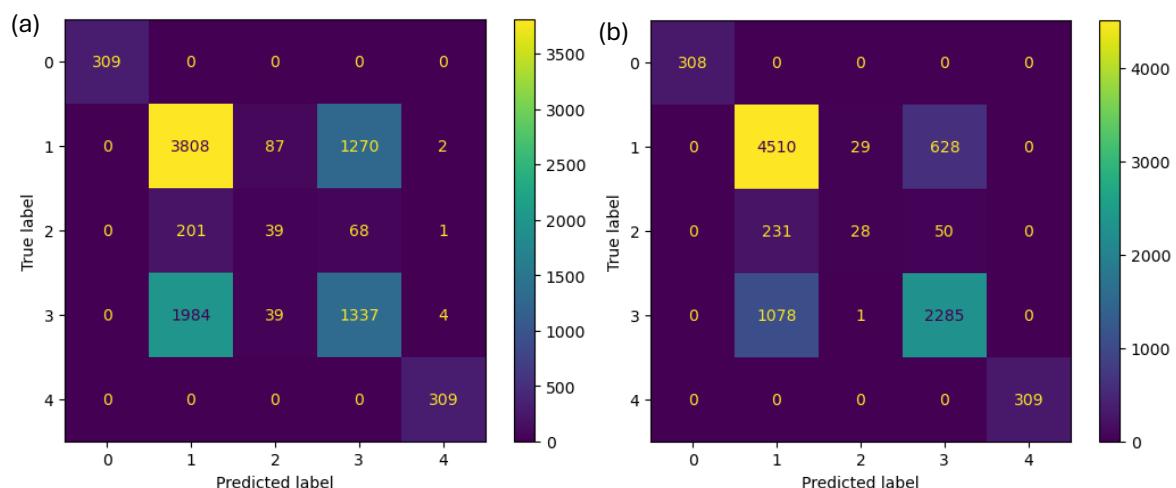


Figure 3. The figures show the results in the sequence labelling task in Task 2. The confusion matrices, and respectively, they are from (a) the LSTM Tagger and (b) the MEMM bi-gram NLM. The models aim to tag the correct SOBIE label for

each word in the given sequence. The number from 0-4 in the matrices represent the SOBIE label class ‘S’, ‘O’, ‘B’, ‘I’, and ‘E’. We focus on the results of class ‘B’ (label = 2) and class ‘I’ (label = 3).

SOBIE Label	S	O	B	I	E	Average (B and I)
LSTM Tagger	1	0.682	0.165	0.443	0.989	0.304
MEMM bi-gram NLM	1.	0.821	0.153	0.722	1.	0.438

Table 4. The table show a comparison the two models used in Task 2 by their F1 scores in each SOBIE label class and the average F1 score of class ‘B’ and class ‘I’. Overall, the MEMM bi-gram NLM performs better in the propaganda snippet detection, which is identifying the correct part of a sentence that should be the propaganda span.

The LSTM tagger gives a far better performance than the MEMM bi-gram NLM does in sequence classification task (Figure 4 and Table 5). The former has an average F1 score 0.23 whereas the latter has an average F1 score 0.09. We can see that many sequences are classified as class “causal_oversimplification” (class label = 6) and none in class “exaggeration_minimisation” is being correctly detected. On the contrary, the LSTM tagger has true positive predictions in every class, displaying a more consistent performance in each class.

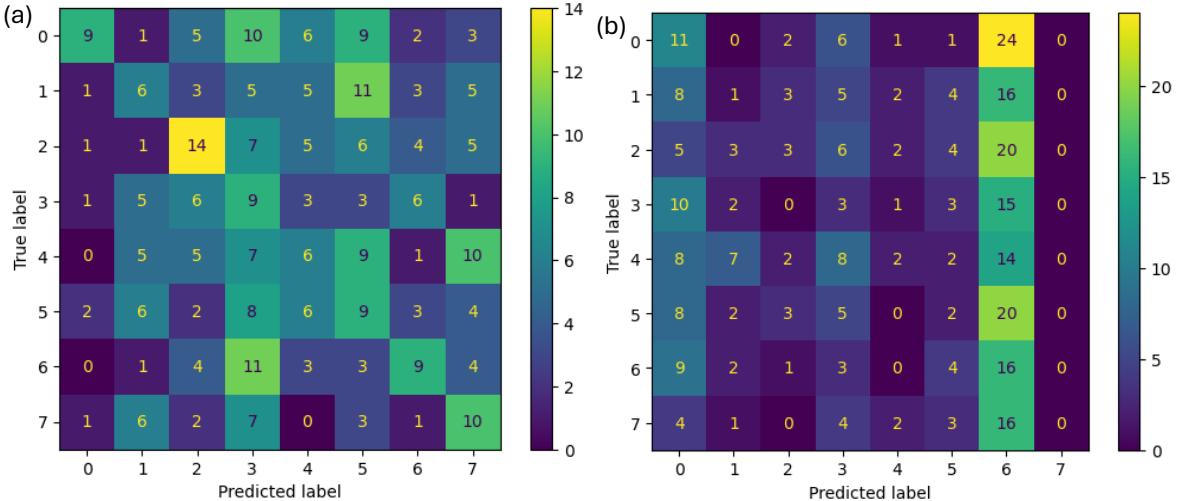


Figure 4. The figures show the results in the sequence classification task in Task 2. The confusion matrices, and respectively, they are from (a) the LSTM Tagger and (b) the MEMM bi-gram NLM. . The numbers from 0-7 represents each of the propaganda technique classes (see Table 2). The LSTM tagger has a far better performance than the MEMM bi-gram NLM model.

Propaganda Technique (Assigned label)	0	1	2	3	4	5	6	7	Average
LSTM Tagger	0.3	0.171	0.333	0.784	0.156	0.194	0.281	0.278	0.237
MEMM bi-gram NLM	0.208	0.035	0.105	0.081	0.075	0.063	0.182	0.	0.093

Table 5. This table show a comparison of the results of the sequence classification task in Task 2. The results are shown by their F1 score in each of the propaganda technique classes, and their macro average F1 score.

Overall, the LSTM tagger has a slightly worse performance than the MEMM bi-gram NLM in propaganda snippet detection, and a far better performance in propaganda technique classification.

4. Discussion and Further Work

Much improvement can be made for this study. Firstly, when creating the bag-of-words representation, this study does not consider the unknown words (i.e. Out-of-vocabulary (OOV) words). For future work, we can create an n-gram probability dictionary and accordingly create an “UNK” tag as representation for rare words that has small unigram probability and a “DISCOUNT” tag as a representation for rare n-grams. Secondly, in Task 2, this study simplifies the process of sequence-to-sequence mapping by tokenising text by words. Different tokenisation can change how we map sequential relationships within text and further give different performance in natural language processing tasks (Choo, 2023) (Yogish, 2019). For future work, we can apply different text preprocessing techniques and observe how they could give different results using the same models and algorithm. Thirdly, the models used for completing Task 1, can be integrate in the pipeline used in Task 2 completion. When given a sentence, MEMM bi-gram NLM generates a SOBIE label with a random choice of “S” label. For future study, the sequence tagger model can load the sequence classifier model for an initial suggestion of the propaganda technique class and could potentially give a more accurate sequence labelling prediction. Fourthly, in Task 2, this study simplifies the conditional probability calculation when applying MEMM technique in the NLM. For future work, we can take the whole sequence observation into account when finding the probability of the current state (Zhu, 2009). Finally, the model training in Task 2 uses only training data; an improvement of the study and be done through using machine learning technique for overfitting prevention (e.g. early stopping, cross validation).

5. Conclusions

In summary, this study provides an easy solution to solve a complicated natural language processing task. Similar to NER task, sequence labelling can be challenging due to the nature of semantic ambiguity in languages (Ratinov, 2009, June); and it also involves multiple subtasks such as sequence labelling task like POS tagging, or sequence classification task like sentiment analysis, to support the learning in the main sequence labelling task (Li, 2020) (Pakhale, 2023). This study presents a novice labelling system, SOBIE labelling system, and simple neural network models that can resolve the challenging task with a decent performance.

Bibliography

- Allison, B. G. (2006). *Another look at the data sparsity problem*. . In Text, Speech and Dialogue: 9th International Conference, TSD 2006, Brno, Czech Republic, September 11-15, 2006. Proceedings 9 (pp. 327-334). Springer Berlin Heid.
- Alshammari, N. &. (2021). *The impact of using different annotation schemes on named entity recognition*. . Egyptian Informatics Journal, 22(3), 295-302.
- Alzaidy, R. C. (2019). *Bi-LSTM-CRF sequence labeling for keyphrase extraction from scholarly documents*. . In The world wide web conference (pp. 2551-2557).
- Balwant, M. K. (2019, July). *Bidirectional LSTM based on POS tags and CNN architecture for fake news detection*. . In 2019 10th International conference on computing, communication and networking technologies (ICCCNT) (pp. 1-6). IEEE.
- Boulesteix, A. L. (2014). *Machine learning versus statistical modeling*. . Biometrical Journal, 56(4), 588-593. .

- Byeon, W. B. (2015). *Scene labeling with lstm recurrent neural networks*. . In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3547-3555).
- Chen, S. Z. (2024). *Multi-task learning in natural language processing: An overview*. ACM Computing Surveys, 56(12), 1-32.
- Choo, S. &. (2023). *A study on the evaluation of tokenizer performance in natural language processing*. . Applied Artificial Intelligence, 37(1), 2175112.
- Du, K. L. (2013). *Neural networks and statistical learning*. . Springer Science & Business Media.
- Dwarampudi, M. &. (2019). *Effects of padding on LSTMs and CNNs*. . arXiv preprint arXiv:1903.07288.
- Ergen, T. &. (2017). *Online training of LSTM networks in distributed systems for variable length data sequences*. . IEEE transactions on neural networks and learning systems, 29(10), 5159-5165.
- Goyal, R. (2021, October). *Evaluation of rule-based, CountVectorizer, and Word2Vec machine learning models for tweet analysis to improve disaster relief*. . In 2021 IEEE Global Humanitarian Technology Conference (GHTC) (pp. 16-19). IEEE. .
- Huang, Z. X. (2015). *Bidirectional LSTM-CRF models for sequence tagging*. . arXiv preprint arXiv:1508.01991.
- Li, J. S. (2020). *A survey on deep learning for named entity recognition*. IEEE transactions on knowledge and data engineering, 34(1), 50-70.
- Limsopatham, N. &. (2016, December). *Bidirectional LSTM for named entity recognition in Twitter messages*. . In Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT) (pp. 145-152).
- Ma, X. &. (2016). *End-to-end sequence labeling via bi-directional lstm-cnns-crf*. arXiv preprint arXiv:1603.01354.
- Mahadevaswamy, U. B. (2023). *Sentiment analysis using bidirectional LSTM network*. Procedia Computer Science, 218, 45-56.
- McCallum, A. F. (2000, June). *Maximum entropy Markov models for information extraction and segmentation*. . In Icml (Vol. 17, No. 2000, pp. 591-598).
- Murphrey, Y. L. (2004). *Neural learning from unbalanced data*. . Applied Intelligence, 21(2), 117-128.
- Ng, A. &. (2001). *On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes*. Advances in neural information processing systems, 14.
- Olson, M. W. (2018). *Modern neural networks generalize on small data sets*. . Advances in neural information processing systems, 31.
- Pakhale, K. (2023). *Comprehensive overview of named entity recognition: Models, domain-specific applications and challenges*. . arXiv preprint arXiv:2309.14084.
- Pasini, A. (2015). *Artificial neural networks for small dataset analysis*. . Journal of thoracic disease, 7(5), 953.
- Patel, A. &. (2021, June). *Fake news detection on reddit utilising countvectorizer and term frequency-inverse document frequency with logistic regression, multinomialnb and support vector machine*. . In 2021 32nd Irish signals and systems confere.
- Prechelt, L. (2002). *Early stopping-but when?*. In Neural Networks: Tricks of the trade (pp. 55-69). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Ratinov, L. &. (2009, June). *Design challenges and misconceptions in named entity recognition*. In Proceedings of the thirteenth conference on computational natural language learning (CoNLL-2009) (pp. 147-155).

- Rish, I. (2001). *An empirical study of the naive Bayes classifier*. . In IJCAI 2001 workshop on empirical methods in artificial intelligence (Vol. 3, No. 22, pp. 41-46).
- Sherstinsky, A. (2020). *Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network*. Physica D: Nonlinear Phenomena, 404, 132306.
- Sun, J. L. (2022). *Deep learning-based methods for natural hazard named entity recognition*. . Scientific reports, 12(1), 4598.
- Xue, J. H. (2008). *Comment on “on discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes”*. . Neural processing letters, 28, 169-187.
- Yao, Y. R. (2007). *On early stopping in gradient descent learning*. . Constructive approximation, 26(2), 289-315.
- Yogish, D. M. (2019). *Review on natural language processing trends and techniques using NLTK*. . In Recent Trends in Image Processing and Pattern Recognition: Second International Conference, RTIP2R 2018, Solapur, India, Decem.
- You, Y. H. (2019, November). *Large-batch training for LSTM and beyond*. . In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (pp. 1-16).
- Zhou, C. S. (2015). *A C-LSTM neural network for text classification*. arXiv preprint arXiv:1511.08630.
- Zhu, J. &. (2009). *Maximum Entropy Discrimination Markov Networks*. Journal of Machine Learning Research, 10(11).