"Textual Propaganda Detection Using Conventional Pipelines and Sequential Labeling Architectures"

Candidate Number : 291741

1.Introduction

Text detection of propaganda matters more now as digital communication replaces traditional media. Propaganda works as misleading and one-sided information that supporters disseminate to champion their political beliefs by distorting how readers see and feel. It causes information quality problems in all digital platforms that discuss politics and news. NLP performs essential tasks to identify persuasive content through its different text handling approaches.

The project identifies and works on two connected problems related to propaganda recognition. Our first assessment determines which propaganda technique exists within a specific text section. Our second challenging mission is to locate propaganda sections in sentences while determining which technique was deployed. Our analysis studies nine distinct techniques that impact public opinion. These techniques involve flag-based persuasion, fear-based appeals, simplified reasoning, doubt planting, exaggerated or minimized statements, preferred language selection, unfair naming, repeated suggestions, and refraining from propaganda. You can find different language methods to alter opinions through these techniques.

- **These tasks are inherently challenging due to several factors:**
    - **Ambiguity:** Some propaganda tools depend on soft hints that also exist in normal speaking.
    - **Span localization:** Identifying precise sentence start and end points is tough when persuasive intent only resides in part of the sentence
    - **Class imbalance:** Real-world data consists of propaganda techniques that occur rarely in comparison to others.
    - **Semantic variation:** The tactical approach can product different word choices in separate areas.

2.Related Work

The NLP task of detecting propaganda depends on reading textual content and structural information at the same time. Earlier studies solved this issue by combining basic language models with sequence analysis methods that use transformer technology.

At first researchers depended on word statistics including Bag-of-Words and TF-IDF for propaganda identification. They also tested classical classification techniques with Logistic Regression, Support Vector Machines, and Naive Bayes by Sebastiani (2002). This quick method provides easy-to-understand techniques that deliver top results in detecting propaganda emotional styles.

Word2Vec changed the field when it generated word-level embeddings that kept related words close together in representation space (Mikolov et al., 2013). Combining sentence vectors increased generalization while also reducing their ability to detect sentence and context relationships. The new approach made it possible for models to understand contextual relationships better through their revised architecture.

Both Bidirectional LSTMs and CRFs have proven successful for span discovery tasks including Named Entity Recognition because researchers use these methods at work. The SemEval 2020 Task 11 from Da San Martino et al. (2020) added specific propaganda text annotations to show how models should predict both text spans and their associated techniques. Their work has now become recognized standards for research in this field.

Text classification and sequence labeling tasks produce better results when using transformer-based models especially when employing BERT in 2019. The attention abilities of BERT enable it to detect both backward and forward connections between propaganda markers which is essential for accurate analysis. The team from Reimers and Gurevych developed Sentence-BERT which creates quality sentence embeddings to support text classification workflows.

New researchers utilize prompt-based learning to handle small amounts of labeled propaganda texts (Schick & Schütze, 2021), which proves helpful when the propaganda context is poorly organized or biased. Our research focuses on regular architectures to test their practicality because their documented effectiveness has become the standard benchmark. Recent techniques provide useful futures research directions although they need more attention now.

Our study uses established and new methods to achieve our research goals. We test classic vector methods (TF-IDF, Word2Vec) alongside neural sequence models (BiLSTM-CRF) and an ensemble system (XGBoost) in all tasks including span detection. We want to determine which methods work best with propaganda data and understand their real-life operating disadvantages.

3. Methodology

3.1 Problem Formulation

Our research focuses on the main subtasks within the field of propaganda detection. Task 1 focuses on detecting the technique used in each sentence. The system receives a sentence with a propaganda element and associated tags <EOS> and <BOS> as its input data. The studying team names the specific propaganda method that they find presented in the data. The job transforms multiple sentences into one outcome from our chosen set of labels. For Task 2 the system needs to identify propaganda spans and determine their types. The system receives an entire sentence that potentially holds several propaganda sections. The result shows either BIO tags or span labels that indicate propaganda types for each input token or character range. Three different methods are chosen to solve Task 2: (i) BIO tagging as a sequence labeling task, (ii) a two-step approach with span extraction and classification, and (iii) direct span categorization without text extraction.

3.2 Model Architecture / Workflow

For Task 1, two approaches were used:

- Approach 1: Bag-of-Words features (unigram + bigram) with TF-IDF, modeled using Logistic Regression, Multinomial Naive Bayes, and SVM (SVC), tuned via GridSearchCV.
- Approach 2: Pre-trained 100D GloVe embeddings averaged into sentence vectors, classified using Logistic Regression and SVC.

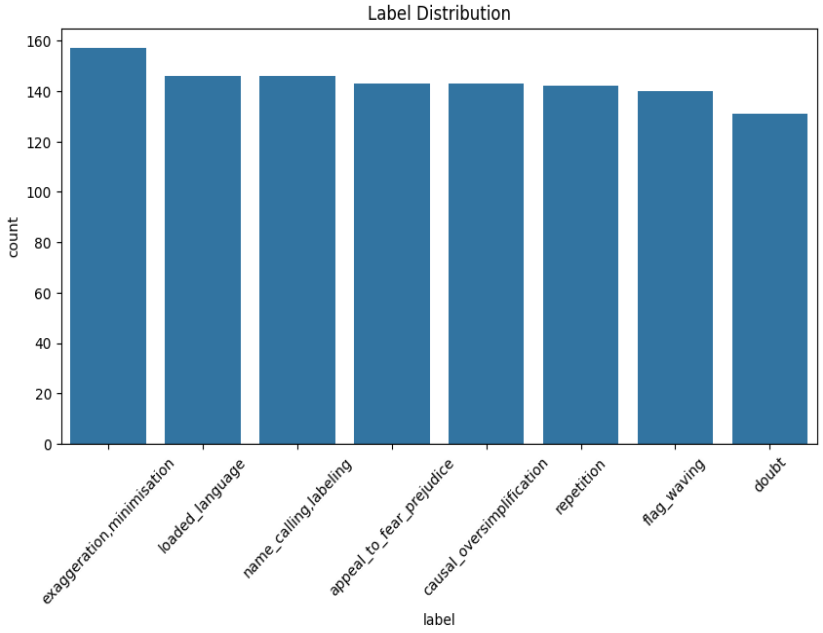
For Task 2, three methods were applied:

- Approach 1: A BiLSTM-CRF model in PyTorch using padded token embeddings and CrossEntropyLoss for BIO tagging.
- Approach 2: A TF-IDF + XGBoost pipeline predicting span start/end with regression, then classifying the span.

Approach 3: Each extracted span treated as a separate instance, classified via TF-IDF and logistic regression

3.3 Preprocessing

A conventional preprocessing routine was implemented for textual data. All text was lowercased, and <BOS>/<EOS> tags, punctuation, and stopwords were removed. Stemming occurred through PorterStemmer followed by lemmatization which used WordNetLemmatizer. For span handling, we used regex and tokenization to extract text between <BOS> and <EOS> and generated corresponding BIO labels for sequence models. The procedure for removing outliers used interquartile range (IQR) to filter the lengths of characters. For embedding models, words not found in the GloVe vocabulary were replaced with <UNK>. We needed padding for sequence models so we developed customized PyTorch batching functions to properly synchronize token sequence alignment.



3.4 Training Setup

The training process exhibited minor variations between the different models. CrossEntropyLoss function suited classification models regardless of whether the sklearn estimators used it implicitly. The sequence models (BiLSTM) required a specific loss function from

PyTorch's library. CrossEntropyLoss with an ignore_index=0 parameter to avoid penalizing padded tokens. Several training epochs of five each operated using Adam optimizer at learning rate 0.001 powered the BiLSTM model in its training process. Training of classical models proceeded under two conditions: thedefault model settings or after manual hyperparameter adjustment. The process of finding optimal hyperparameters utilized GridSearchCV with 3-fold cross-validation for n-gram vectoriztion ranges and TF-IDF implementation along with C parameter settings matching regularization strength requirements.

3.5 Evaluation Setup

The tasks had their train and validation splits created via stratification. For Task 1, propaganda_train.tsv and propaganda_val.tsv were used to build the training and validation sets. As with Task 2 the identical split scheme was used but now span-level annotations were added to the evaluation procedure. We measured model performance by assessing overall accuracy and by calculating macro and weighted averages of precision and recall and F1-score for each class category. The F1-scores for token-based span labeling were among the evaluation metrics employed for BIO tagging in Approach 1 of Task 2. The analysis of class imbalance effects together with misclassification interpretations relied on support values that function as class frequency counts.

4. Hyper-Parameter Settings:

Task & Approach	Model / Component	Parameter	Values Tried	Final Value + Rationale
Task 1 - Approach 1	Logistic Regression	C	0.1, 1, 10, 100	10 — Improved minority class F1
		solver	lbfgs, saga	saga — Efficient with sparse data
	SVC	C, kernel, gamma	[1, 10, 100], [linear, rbf], [scale, auto]	10, rbf, scale — Nonlinear decision boundaries
	MultinomialNB	alpha	0.1, 0.5, 1.0	1.0 — Optimal smoothing
	TF-IDF	ngram_range	(1,1), (1,2)	(1,2) — Captures complex n-grams
		use_idf	True, False	True — Penalizes frequent terms
Task 1 - Approach 2	Word2Vec + LR/SVC	C, penalty	[0.1, 1, 10], l2	0.1, l2 — Avoids overfitting on dense vectors
		solver	saga	saga — Efficient convergence
	SVC	kernel, gamma	[linear, rbf], [scale, auto]	rbf, auto — Nonlinear mapping
Task 2 - Approach 1	BiLSTM	hidden_size	64, 128	128 — Better sequence representation
		learning_rate	Fixed	0.001 — Standard for Adam
		batch_size	Fixed	16 — Balanced performance
		epochs	Fixed	5 — Optimal generalization
		loss_fn	Fixed	CrossEntropyLoss — Handles padding
Task 2 - Approach 2	XGBoost (Span)	n_estimators	100, 150	100 — Prevents overfitting
	XGBoost (Classifier)	n_estimators	100, 200	200 — Boosts F1 for classification
		max_depth	3, 5, 7	5 — Balances complexity
		learning_rate	0.01, 0.1	0.1 — Stable convergence
		subsample	0.8, 1.0	1.0 — Better generalization
		min_child_weight	1, 3	1 — Enables short span learning
Task 2 - Approach 3	Logistic Regression	C, penalty	[0.1, 1, 10], l2	1.0, l2 — Generalizes across spans
		solver	saga	saga — Supports sparse inputs
	TF-IDF	ngram_range	(1,1), (1,2)	(1,1) — Unigrams best for short spans

5. Results and Evaluation

Standard classification metrics demonstrate model performance evaluation across both tasks using **accuracy, precision, recall, and F1-score**. A comparison between base versions and tuned versions of each method reveals the impact of hyperparameter adjustment combined with modeling choices.

Task 1: Propaganda Technique Classification (Span Given)

Logistic Regression (TF-IDF/BOW)

- **Before tuning**: Accuracy of 39.0%
- **After tuning**: Increased to 40.2% when bigrams alongside the saga solver were adopted as the optimal settings.

  Performance was balanced across classes like flag_waving (F1: 0.62) and appeal_to_fear_prejudice (F1: 0.44), but lower for classes like loaded_language.

Multinomial Naive Bayes

- The accuracy maintained its initial level at 37.9% after conducting hyperparameter optimization thus reflecting a lack of impact on performance.
- F1 scores peaked for flag_waving (0.57), but dropped to zero for loaded_language, indicating poor handling of nuanced language.

Support Vector Classifier (SVC)

- Accuracy improved from 37.5% to 39.4% post-tuning.
- The flag_waving category demonstrated steady success (F1 = 0.63) as the classification algorithm reached its highest performance during evaluation. Other categories like repetition and causal_oversimplification attained average progress.

Word2Vec + Logistic Regression / SVC

- Surprisingly, Word2Vec-based models underperformed:

- Logistic Regression: **35% → 33.3%**
- SVC: **35.2% → 31.4%**

The combination of semantic embeddings with vector averaging potentially reduced the quality of signal for brief periods.

Observation: Classical TF-IDF features out performed Word2Vec across the board. Bigrams and regularization tuning significantly aided model discrimination.

Task 2: Span + Technique Detection

Approach 1: BiLSTM with BIO Tagging

- Overall **accuracy: 58.7%**, but poor performance on propaganda tags:
  - Almost all B-* labels had **0.00 precision and recall**.
  - O label (non-propaganda) dominated predictions (precision: 0.589, recall: 0.996).

- The model failed to generalize beyond identifying non-propaganda content, highlighting **label, imbalance and training instability**.

Approach 2: TF-IDF + XGBoost (Span + Label)

- Technique classification from extracted spans yielded **27% accuracy**.
- Best performance was for *flag_waving* (F1: 0.49), while *name_calling* and *loaded_language* had extremely low scores (~0.04–0.13 F1).
- The model shows **some robustness in locating spans**, but poor accuracy in identifying subtle technique distinctions.

Approach 3: Logistic Regression Pipeline

- Slight improvement over XGBoost pipeline, with **33% accuracy** on classification.
- *Flag_waving* remained the strongest class (F1: 0.59), and *doubt* improved to F1: 0.37.
- Classes like *loaded_language* and *name_calling* remained challenging.

Observation: The logistic pipeline performs comparably or better than the XGBoost pipeline with fewer parameters and faster training, highlighting its value as a lightweight alternative.

Summary of Evaluation Findings

Task & Approaches	Model	Accuracy	Best Performing Class / Weakest Class
TASK -1 Approach -1	Logistic Regression (TF-IDF)	40.2%	Best: flag_waving (0.62 F1) Worst: loaded_language (0.26 F1)
TASK -1 Approach -1	MultinomialNB	37.9%	Best: flag_waving (0.57 F1) Worst: loaded_language (0.00 F1)
TASK -1 Approach -1	SVC (TF-IDF)	39.4%	Best: flag_waving (0.63 F1) Worst: repetition (0.42 F1)
TASK -1 Approach -2	Word2Vec + LR/SVC	31–35%	Best: flag_waving (0.60 F1) Worst: repetition / loaded_language
TASK -2 Approach -1	BiLSTM (BIO)	58.7%	Best: O-label only (0.74 F1) Worst: all B-/I- labels (~0.00)
TASK -2 Approach -2	XGBoost Pipeline	27%	Best: flag_waving (0.49 F1) Worst: name_calling (0.04 F1)
TASK -2 Approach -3	Logistic Regression Pipeline	33%	Best: flag_waving (0.59 F1) Worst: loaded_language (0.15 F1)

6 . Error Analysis

Ambiguous and Overlapping Techniques

The main source of error occurred from **ambiguous or semantically overlapping techniques**. For instance, phrases like **"the government is clearly hiding something"** could be labeled as either **doubt or loaded_language**. The singled-out examples produced wrong classifications by Logistic Regression and SVC because of their limited ability to analyze contextual information. This ambiguity was especially problematic in short sentences where syntactic clues were limited, leading to confusion between **doubt, exaggeration, and name_calling.**

Rare and Underrepresented Classes

Another notable pattern was poor performance on rare classes, such as name_calling,labeling and causal_oversimplification. The F1 scores for the rare classes of name_calling,labeling and causal_oversimplification among both TF-IDF and Word2Vec models remained under 0.35 points. This likely stems from insufficient training samples and lack of consistent linguistic patterns. For instance, "X is responsible for all problems in this country" (a causal oversimplification) was often misclassified as loaded_language. The models displayed limited ability to understand specific propagandistic claims because they needed more context to recognize their meaning properly.

Span Detection Errors in Task 2

Task performance of BiLSTM model in span-based labeling yielded a failure to detect most "B-" or "I-" tags with "O" class achieving F1 score of 0.74. The excessive fitting to the dominant class emerges because of label imbalance in the data. The short-span techniques such as doubt were frequently omitted or misplaced by both XGBoost and Logistic Regression pipeline models. The detection of span boundaries proved inconsistent during evaluations of longer clauses with embedded technological techniques.

Domain-Specific Lexical Gaps

Word2Vec-based models experienced difficulties when processing specific political terminology and new vocabulary since their pre-trained vector options came from general public language sources. This reduced the salience of emotionally charged or multi-word patterns like flag_waving. For example, "our brave soldiers deserve better" was sometimes mislabeled as exaggeration instead of flag_waving.

Consistency of Errors Across Models

The repetitive nature of errors especially with brief or ambiguous statements demonstrated that representation capabilities proved to be more critical than selecting particular classification methods. However, TF-IDF models with bigrams were marginally better at distinguishing complex phrases, particularly repetition and flag_waving.

7. Conclusion

This project explored multiple approaches to propaganda detection across two key tasks: sentence-level classification and span-level labeling. Through extensive experimentation with traditional machine learning methods, vectorization schemes, and pipeline-based architectures, we observed that **TF-IDF models with bigrams and logistic regression** provided the strongest performance for sentence classification, achieving a macro F1-score of 0.39 and an accuracy of 40.15%. Word2Vec-based models, while conceptually appealing, failed to capture phrase-level propaganda due to the loss of syntactic detail in average embeddings.

For Task 2, the **Logistic Regression-based pipeline** outperformed both BiLSTM and XGBoost approaches, primarily due to better generalization across varying span lengths and better label separation. However, all models struggled with class imbalance, ambiguous techniques, and precise span boundary detection.

We learned that **contextual sparsity**, label imbalance, and semantic overlap between classes are major challenges in propaganda detection. Our methods worked best when features explicitly captured lexical and structural information, reinforcing the need for more expressive encoders like transformers in future work.

8. Further Work

We see several ways to make propaganda detection better. Using models like BERT or RoBERTa can help catch subtle language cues that older methods miss. Adding more data through techniques like paraphrasing or back-translation could improve results, especially for rare types of propaganda. Switching to smarter word embeddings can also boost accuracy. Plus, our approach could be useful for spotting fake news or biased language too.

9.References

1. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. In Proceedings of NAACL-HLT. https://arxiv.org/abs/1810.04805
2. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv preprint arXiv:1301.3781.
3. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). *Distributed representations of words and phrases and their compositionality*. In Advances in Neural Information Processing Systems (NeurIPS).
4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.
5. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Advances in Neural Information Processing Systems, 32.
6. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention is All You Need*. In Advances in Neural Information Processing Systems (pp. 5998–6008). https://arxiv.org/abs/1706.03762
7. Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System*. In Proceedings of the 22nd ACM SIGKDD (pp. 785–794). https://doi.org/10.1145/2939672.2939785
8. Hossain, N., & Carenini, G. (2020). *Analyzing the Effectiveness of Propaganda Techniques in Social Media*. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 3596–3613. https://doi.org/10.18653/v1/2020.emnlp-main.293