Exploring Semi-Supervised Learning and Transformers for Fake News Detection

Group Members:
Devang Chaudhary
Nikash Prakash
Venkat Kannan

Problem Description

We will be using semi-supervised techniques, along with clustering-based pseudo-labelling, to create a model that identifies whether an article is fake news or not.

Fake news detection is something that all of us are deeply passionate about. We feel that news has become more commercialized and is being used to push agendas rather than speak the truth. We think this way for both aisles of the political spectrum. However, we also acknowledge the importance of having a free and effective media ecosystem to promote democracy and growth. We believe that one way to help solve this issue is to identify fake news so that news companies can be held accountable and democracy can be strengthened. Natural Language Processing is one of the main ways for us to detect such biases, as we can do this automatically.

The input that our model will receive is the text of an article headline. The model (transformer-based) will then process this input and output a binary label, indicating whether the article is fake news or not.

The most significant challenge involved in this task deals with the training datasets of the model. There is a need for more extensive amounts of labeled training data, as the currently labeled datasets are not comprehensive enough. In addition to this, the ever-changing and evolving nature of news makes it so that we need to reflect this by having an updated and current/recent training dataset. Otherwise, the model won't be able to adapt and accurately identify more recent news articles. Another challenge involves the transformer models themselves, as there is a need to create special transformer models suited for and trained extensively for this task, instead of being fine-tuned.

We decided to work on the challenge that involves the labeled training datasets not being comprehensive enough. We used an unlabelled dataset and created pseudo-labels for it to

increase the size of our overall training dataset. This approach also mitigates the issue of not having current/recent data as it does not require current/recent data to be labeled.

As for task allocations, we did not specifically allocate tasks to anyone in particular. Instead, it was a group effort. We would all work on the project together in meetings. We would also live share the code using VS Code live share. During the meetings, some people may work on different tasks, but that was usually decided during the meeting and there were no pre-divisions. The divisions also involved equal work. We all spent a similar amount of time and effort on the project.

Related Work

The field of fake news detection has been an integral part of Natural Language Processing and subject to extensive research. Over time, many different approaches have been tried to detect such biases as the field has evolved.

The pioneering and traditional approaches used machine learning techniques, such as logistic regression, support vector machines, and random forests, on linguistic features (Rodrigo-Ginés, Francisco-Javier, et al., 2023). The models relied on lexical features such as n-grams, syntactic features such as part of speech, and semantic features such as named entity recognition. For example, Krestel, Wall, and Nejdl (2012) employed the vector space model from information retrieval, using 15 years of speeches from the German Bundestag and with term weights based on TF-IDF scores. They were then able to detect fake news/political leanings in media content. However, we still wanted to find a way to increase the performance of such models.
As a result, with their recent popularization, deep learning models started being used to detect fake news. Deep learning models have two main advantages, they can automatically learn feature representations from text and are more capable of modeling the sequential structure of a sentence (Rodrigo-Ginés, Francisco-Javier et al., 2023). LSTM RNNs were primarily used as they can model long-term dependencies in a sentence by using internal memory, in addition to being able to In 2017, Rashkin et al. (2017) showed that an LSTM-type RNN outperforms other classical machine learning models such as Naïve Bayes or Max-Entropy. A similar analysis was done by Baly, Da San Martino et al. (2020), obtaining better results. However, a disadvantage of RNN models is that the classification is done based on the last hidden state. This can be problematic for long sentences as the weights from the different input sequences have to be correctly represented in the last state.

Due to this disadvantage, transformers have now become the current State-of-the-art in detecting fake news. The main difference between Transformers and RNNs is that Transformers do not have an internal state. Instead, Transformers use the self-attention mechanism to model the

sequential structure of a sentence (Rodrigo-Ginés, Francisco-Javier, et al., 2023). Because of this, Transformers have been shown to outperform RNNs in modeling the sequential structure of a sentence. In fact, transformer models have begun to displace models based on linguistic features and RNNs, since they generally obtain better results for media bias detection (Baly, Da San Martino et al., 2020).

However, there is still a lot of work to be done for fake news detection. One of the potential areas of improvement that we have identified involves the training datasets of these transformer models. There is a need for more extensive amounts of labeled training data, as the currently labeled datasets are not comprehensive enough. In addition to this, the ever-changing and evolving nature of news makes it so that we need to reflect this by having an updated and current/recent training dataset.

We will tackle this problem in a new way by using semi-supervised learning. Specifically, we will utilize the cluster-then-label technique. This will enable us to use the current limited amount of training data and clusters to generate pseudo-labels for a more extensive and bigger set of unlabeled data. As such, this will allow our model to use previously untapped datasets that are not fully labeled thus allowing us to not rely entirely on human-labeled data, limiting the training potential of our model. In addition, this approach will allow us to mitigate the changing nature of news as we can utilize the latest news articles to update our model without needing any human labeling or input. We believe that this will enable us to create a new approach to this task and, at the same time, improve performance compared to previous approaches. We can also extend the idea of an "SSL cluster-then-label + transformer" approach to other NLP classification tasks where there is a lower number of labeled data than wanted. This raises the question of whether there is a need to develop clustering methods that are optimized for SSL (as opposed to unsupervised learning). In short, at a high level, our project provides insights into the potential of using semi-supervised techniques and clustering algorithms to create pseudo-labels along with state-of-the-art transformers to solve a classic supervised binary classification problem.

Methodology

Datasets:

We are using two datasets from the MBIB dataset created by the Media Bias Group: fakenews.csv (labeled, datapoints = 8542) and political.csv(datapoints, size = 17704).

Fakenews.csv (Labeled) Data Points

Id	Text	Label	Dataset_Id
012-4814	Police confirm several shootings in Ottawa PM safe downtown buildings in lockdown	1	12

Political.csv (Unlabeled) Data Points

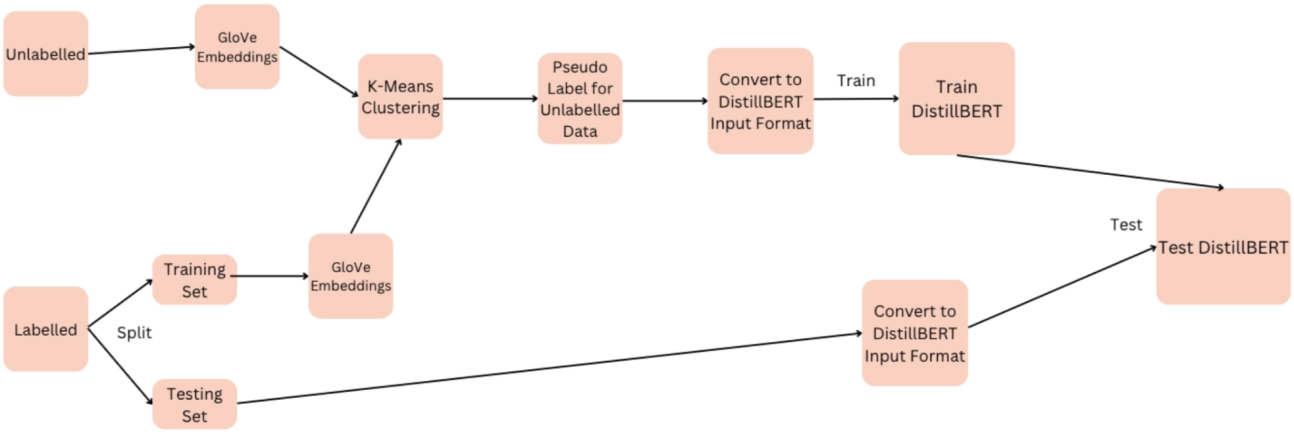
Id	Text	Dataset_Id
029-0000715-2	New documents detail a young girl s account of the night her father and siblings were killed in a gruesome attack	29

Fakenews.csv: This is a binary classification task to predict fake news versus real news (0 = real, 1 = fake). The classes are perfectly split with 4271 data points for each, thus no class imbalance issues and no immediate bias issues. The dataset consists of four columns: Id, Text, Label, Dataset_id (example above). For the purpose of our task, we only considered the Text and Label columns.

Political.csv: This is an unlabelled dataset consisting of various texts from political articles. The dataset consists of three columns: Id, Text, Dataset_Id. There are no immediate biases or focuses of the political text. For the purpose of our task, we only considered text.

The MBIB dataset was created/collected by the Media Bias Group. We downloaded the dataset from their GitHub repository (https://github.com/Media-Bias-Group/MBIB#introducing-mbib---the-first-media-bias-identification-benchmark-task-and-dataset-collection). This dataset is suitable for our task as it provides us with labeled data for our exact classification task along with similar unlabeled news/political articles that can be used for pseudo-label creation. The binary labeling within the dataset allowed us to utilize the dataset directly and completely use it without needing to modify it for our task. The straightforward text format of the dataset made is easy to preprocess and modify as needed to input into the numerous models used throughout this project. In short, the relevance and structure of the dataset makes it very useful for our task.

Our Model:



Below is the information on how our model and preprocessing works.

1. Preprocessing:

    a. Split labeled data into training and testing data (75% to 25% split). All the unlabeled data will be used for training.

    b. Lowercase every word in all datasets and remove all stopwords and non alphanumeric words

    c. Extract the average Glove Embedding of every article in the training data. We do this by averaging the glove embeddings of every individual word that makes up the article.

2. Utilize cluster-than-label on all the unlabeled data and the labeled training data (based on their glove embeddings):

    a. Create 2 clusters using the k-means clustering algorithm

        b. Label the clusters based on the most prevalent label in it (from labeled data). We use ratios to do this (more details below in the clustering section).

    c. Pseudo-label each unlabelled data based on which cluster it belongs to.

3. Train our model on this data (labeled training data and all pseudo labeled [previously unlabeled] data) (in raw text format and not glove embeddings)

    a. Tokenize the input using the DistilBert tokenizer, so that it is in the required input format

    b. Run the input through a DistilBert transformer layer

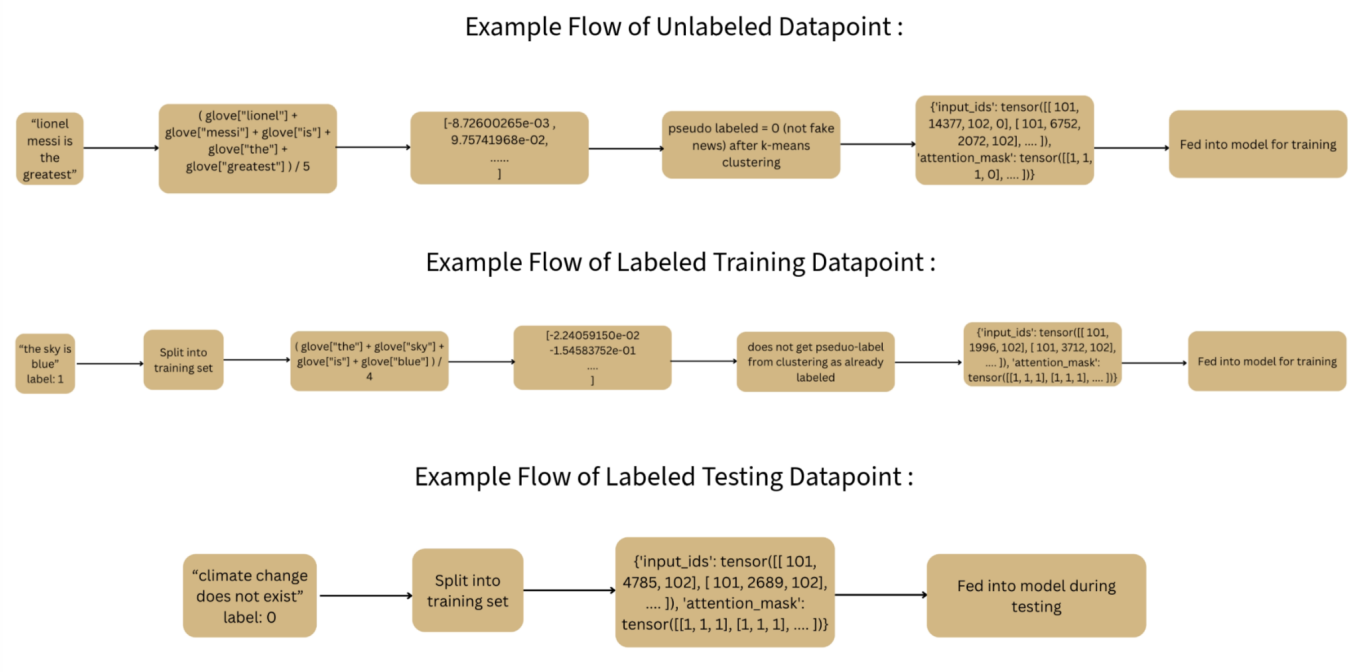
c. This is then connected to a linear layer -> ReLU function -> dropout -> linear layer -> softmax function

d. Adam optimizer and a criterion of cross entropy loss are used for model optimization

e. Utilizing random search to obtain optimal values of hyperparameters (learning rate, optimizer weight decay, dropout rate) using a separate training and validation set. We use a 90% to 10% split for training and validation, respectively.

4. Test our model using the labeled testing data.
5. Evaluate performance and compare with baseline models.

Here are some sample toy cases illustrating the pathway for different input data points:



Clustering:

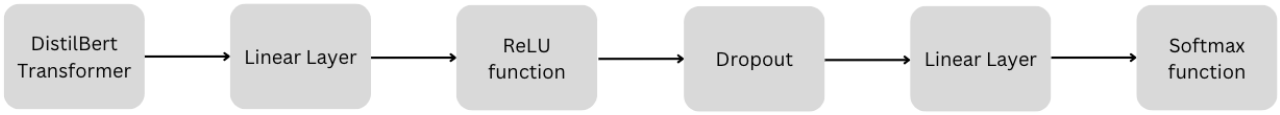
- We are currently using a K-Means clustering algorithm that takes in the 200-length vector sentence-embeddings of both the labeled training dataset and the whole unlabeled dataset as input and produces two clusters. The motivation behind this is clustering group articles based on similarities in their features and semantic properties, while being applicable to both labeled and unlabelled data. The motivation behind using K-Means clustering is its straightforward implementation and efficiency with large datasets (we initially tried Spectral but was too slow and needed too much memory). Once we have generated these

clusters, we then need to label which cluster corresponds to fake news and which one doesn't. So, to do this, we count the number of fake news articles in cluster 0 and the number of fake news articles in cluster 1. Similarly, we count the number of not fake news articles in cluster 0 and the number of not fake news articles in cluster 1. We then calculate ratio A to be the number of fake news articles in cluster 0 divided by the number of fake news articles in cluster 1. We also calculate ratio B to be the number of not fake news articles in cluster 0 divided by the number of not fake news articles in cluster 1. If ratio A > ratio B, we label cluster 0 to correspond to fake news while we label cluster 1 to correspond to not fake news. If ratio B > ratio A, we label cluster 0 to correspond to fake news while we label cluster 1 to correspond to not fake news. This way, we label each cluster based on which category is more prevalent in it, relative to the other cluster.

- From here, for every unlabeled data point, we will pseudo label it with the same value as the cluster label it belongs to.
- We also tried Hierarchical, GMM, DBSCAN, and spectral clustering methods. But, K-Means works best for us in terms of performance and efficiency.

DistilBERT-based Transformer Model:

- We are using a bidirectional transformer based model that takes in the averaged word embeddings (sentence embedding) and tries to predict the label as determined from the previous clustering step. The motivation behind using a bidirectional encoding transformer like Disitlbert is that we want to ensure that it picks up on confusing sentiment from the data. This will allow it to 'understand' the text better to grasp the text's stance (fake news as of now, but political leaning for the final).



- We are currently using DistilBert as our main transformer block. The reason behind using DistilBert is that it is lightweight in comparison to BERT and keeps most of BERT's performance in stance detection. We first feed our input into the DistilBert block and then pass the output of the transformer to two fully connected layers and one dropout layer in between. We have 768 as the input for the first linear layer, and 768 as the output. We then apply the ReLU function to the output of the first layer. After this, we regularize the output from the first dense layer through the dropout layer to avoid overfitting. We are currently using a 30% dropout. We will use cross validation to get an optimal value for the dropout percentage. From here, we pass the output to the second linear layer. So, for the second layer there are 768 inputs and 2 output nodes. We then run this through a softmax function. This normalizes the 2 values into a probability distribution, which will

give us the probability that the initial model input is fake news and the probability that the initial model input is not fake news.

- We are optimizing our model using the Adam optimizer and a criterion of cross entropy loss.
- We can now pass in new text inputs into the model and predict whether the input is fake news or not. The model predicts an output classification based on whether the probability value of it being fake news or the probability value of it not being fake news is higher.

Experiments

In running our model, we split the labeled data set into a 75% train and 25% testing split from the labeled dataset. After merging the pseudo-labelled data with the rest of the labeled training data, we then split again with 90% kept for training and the other 10% for validation. A SSL task is going to deal with a small number of labeled-data points and a large number of unlabeled-data points. As such, it is very important that our model generalizes well to unseen data. Thus, we used a validation set and selected appropriate hyperparameters. For the hyperparameters (learning rate, dropout, and optimizer weight decay), we used random search with a dedicated split. We set the batch size to be 64, as it seemed like a balanced batch size, given the volume of data it was enough to speed up the optimization step with ADAM. We ended up using a learning rate of 5e-5, and for dropout rate of 0.75 and optimizer weight decay 1e-5. To make sure that we didn't end up overfitting the model, we implemented regularization technique early stopping with a patience of 5 (runs 5 epochs past 'best' found - based on the validation loss). The model trained for 8 epochs in total, and the best validation loss was at epoch 3, but we used epoch 4 as we saw better results across all metrics than epoch 3. We checked one epoch below and above our best validation loss epoch.

We will be using accuracy, precision, and recall as our three evaluation metrics. Below are the explanations for the usage of the three metrics.

- Accuracy: We can use accuracy to check how many data points were correctly predicted as bias/unbiased, the degree/strength of the bias, and figuring out the specific bias-inducing phrase.
- Precision: Classifying true news as fake is equally as damaging for information dissemination as the opposite, therefore evaluating the precision of our model will inform us the existence of the model's false positives. Tracking precision will ensure that we do not engage in a severe precision/recall tradeoff and predicting more articles are fake than really are.
- Recall: Recall value is the primary value that will inform us how well our model is detecting fake news as fake news. A high recall value indicates our model is

sufficient at reducing false negatives and succeeds in its primary objective of distinguishing fake news from genuine articles

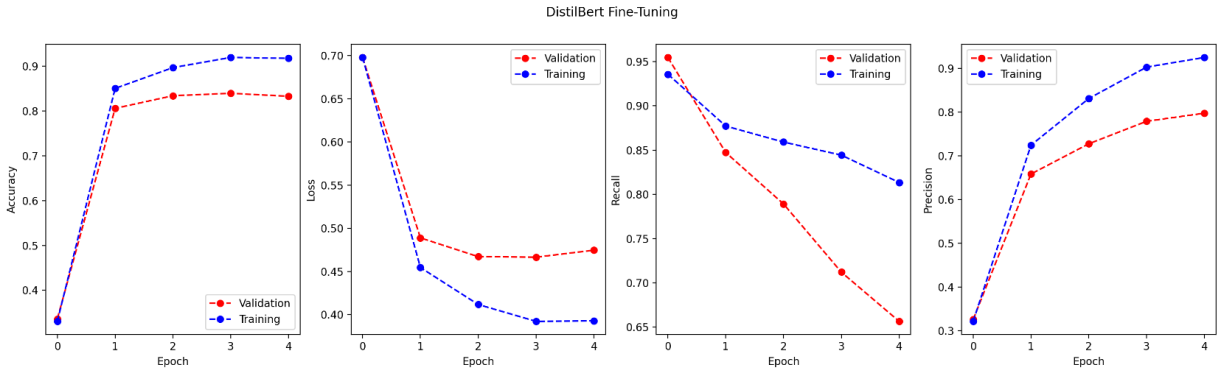
Overall, the above metrics will give us a good insight into the performance of the model and more importantly specifically pinpoint where the model may be lacking.

Model Performance:

We have two baselines:

1. Simple baseline. The simple baseline always predicts an input as fake news.
2. FNN baseline. This is a simple three-layer feedforward neural network
    a. The NN is a three-layer feedforward model with 64 neurons in the first layer and 32 neurons in the second layer. The reason we went with three layers was to ensure our model does not overfit to the data and, at the same time, simple and fast enough to serve as a baseline. The decision for 64 and 32 neurons was based on common NN practices (again did not want to increase and overfit). The total number of trainable parameters was 14977. The first two layers used a ReLu activation function in order to avoid gradient-approaching-zero problem with Sigmoid and used Sigmoid in the final layer as our NN is still a binary classifier. The NN used a standard Adam optimizer (along with its default hyperparameter values), Binary-Cross Entropy loss, 80-10-10 training, validation, test split, and ran on 20 epochs (validation loss was increasing despite lowering training accuracy after 20 epochs thus indicating overfitting).

The graph below shows the performance of our DistilBert based model during the training stage:



The below table contains the performance of the two baseline models on our training data set.

	Accuracy	Precision	Recall
Simple Baseline	0.50	0.50	1
FNN Baseline	0.57	0.62	0.50
DistilBert	0.72	0.63	0.78

As initially expected, the transformer performed much better than the simple baseline with a 22% increase in accuracy, and also better than the FFN baseline with a 15% increase in accuracy. Furthemore, the transformer had a higher precision than both the baselines, even though it was not significantly higher than the FFN-baseline precision. It also had a higher recall than the FFN-baseline. The simple baseline had a trivial recall of 1, so this doesn't affect our results. These better metric performances can be attributed to an increase in true positives, while also having a decrease in false negatives. This better performance can be attributed to the transformer model having a more intensive architecture, greater number of total parameters, and attention mechanism to model the sequential structure of language.

However, our final evaluation metrics could have been better. Below is an analysis on this statement.

Potential sources of errors (i.e. areas of possible improvement in current approach) could occur in how we create pseudo-labels for the unlabelled data. First, we are converting all words into GloVe-embedding and averaging all the GloVe-embeddings within an article. In doing so, we are losing the sequential structure of the words - which, of course, can completely determine the meaning/factuality of the article. Longer articles might be prone to losing key info and shorter articles might not have enough information when averaged. As such, the inputs that are passed in clustering already have a major structural issue thus undermining the effectiveness of the clustering algorithm regardless of how good it is on its own. It may be better to pass it the glove embedding over every word as a list with padding instead of averaging.

In addition, K-Means may not be the most effective as, at a base level, it assumes clusters are spherical and equidistant from each other. In a 200-level GloVe dimensional representation, these assumptions are most likely not true. Also, K-Means sensitivity to outliers can undermine the effectiveness of clusters as outliers can skew the mean of the clusters significantly. Articles that discuss unique issues or have a different style of writing can become outliers. It may be useful to

consult industry research papers to identify additional clustering methods that we have not tried yet. In addition to this, contrastive learning may be able to improve performance in the clustering stage. Contrastive learning contrasts negative and positive pairs, by utilizing the assumption that similar instances should be closer together in an embedding space, while differing instances should be farther apart. This should help with getting a better set of clusters in our clustering stage.

Lastly, DistillBERT is a smaller version of the parent model BERT. Therefore, DistillBERT may not capture all the complexities that BERT could capture - which, when dealing with a task as difficult as detecting patterns within fake news may be necessary. With the real-world significance of this task, even a marginal improvement is vital.

As a closing remark, we are content with the results of our project and learned a great deal about semi-supervised learning and transformers. Thank you for a great class!

Code Link (includes README file)

Here's a link to our github, which includes our code and README file:
https://github.com/NikashPrakash/FakeNewsDetection

https://github.com/NikashPrakash/FakeNewsDetection

Works Cited:

Baly, R., Da San Martino, G., Glass, J., & Nakov, P. (2020). We Can Detect Your Bias: Predicting the Political Ideology of News Articles. In *Proceedings of the 2020 conference on empirical methods in natural language processing* (pp. 4982–4991).

Francisco-Javier Rodrigo-Ginés, Jorge Carrillo-de-Albornoz, Laura Plaza (2023). A systematic review on media bias detection: What is media bias, how it is expressed, and how to detect it, Expert Systems with Applications, Volume 237, Part C, 2024, 121641, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.121641.

Krestel, R., Wall, A., & Nejdl, W. (2012). Treehugger or Petrolhead? Identifying bias by comparing online news articles with political speeches. In *Proceedings of the 21st international conference on world wide web* (pp. 547–548).

Rashkin, H., Choi, E., Jang, J. Y., Volkova, S., & Choi, Y. (2017). Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 2931–2937).

Spinde, Timo (2023). Introducing MBIB - the first Media Bias Identification Benchmark Task and Dataset Collection. https://github.com/Media-Bias-Group/MBIB#introducing-mbib---the-first-media-bias-identification-benchmark-task-and-dataset-collection.

Duarte, J.M., Berton, L (2023). A review of semi-supervised learning for text classification. Artif Intell Rev 56, 9401–9469. https://doi.org/10.1007/s10462-023-10393-8