NewsFactChecker: Bayesian, Kernel, and Transformer Claim Detection

Smyan Sengupta
Sankalp Aswani
Sandeep Salwan

Northeastern University, Boston, MA
{sengupta.sm, aswani.sa, salwan.s}@northeastern.edu

Abstract

The rapid spread of misinformation online poses a major threat to media credibility and informed public discourse. In this work, we investigate three complementary approaches for automated fact-checking: probabilistic modeling via Bayesian inference using Markov Chain Monte Carlo (MCMC), kernel-based classification using Support Vector Machines (SVM), and transformer-based deep learning using a DeBERTa model. Using the LIAR dataset, we train and evaluate all models using standard metrics, including F1 score, accuracy, and AUC. Our DeBERTa model achieves the best performance with an F1 score of 0.56 and ROC-AUC of 0.683, outperforming the TF-IDF + logistic regression baseline. SVM and Bayesian methods, while slightly lower in accuracy, offer interpretability and uncertainty estimation. These results highlight the value of deep contextual embeddings in combating misinformation effectively.

Introduction

The digital age has transformed the way information is produced and consumed. Unfortunately, it has also fueled the rapid spread of misinformation. Distinguishing factual statements from fabricated claims is critical. News articles, often containing nuanced or manipulative statements, can sway public opinion, impact elections, and erode trust in journalism (Wang 2017).

Our goal is to build an AI-powered misinformation detection system capable of evaluating individual claims extracted from articles. This task is formally a binary classification problem: given a claim c, the model outputs whether c is true (1) or false (0). The classification process relies on linguistic understanding, particularly transformer-based embeddings that capture rich contextual meaning from short statements (He et al. 2021).

In this paper, we explore three different approaches for claim verification: Bayesian inference using MCMC, support vector machines (SVM), and deep learning using a DeBERTa model. We evaluate and compare their performance on a labeled dataset.

Background

Prior work in misinformation detection has leveraged probabilistic models, kernel methods, and deep neural architectures. BERT-based models have shown strong results in sentence-level classification (Devlin et al. 2019), and DeBERTa improves on BERT with disentangled attention and relative position encoding (He et al. 2021). Traditional baselines like logistic regression and SVM using TF-IDF remain common for benchmarking (Kleinbaum and Klein 2002).

Logistic Regression

Logistic regression is a linear classifier that models the probability of a binary outcome using the logistic (sigmoid) function. It is trained using cross-entropy loss and is commonly applied to text classification with features like TF-IDF (Kleinbaum and Klein 2002). **Term Frequency–Inverse Document Frequency (TF-IDF)** is a statistical technique used to measure the importance of a word in a specific document relative to a collection of documents (the corpus). In the context of misinformation detection, TF-IDF helps identify which words in a claim are particularly distinctive or meaningful compared to other claims in the dataset (Kleinbaum and Klein 2002).

Naive Bayes Classifier

Naive Bayes is a probabilistic model based on Bayes' Theorem, assuming feature independence. It performs surprisingly well in high-dimensional settings like text, where word presence can be treated as independent binary events (Kleinbaum and Klein 2002).

Support Vector Machine (SVM)

SVMs are non-probabilistic margin-based classifiers that aim to find the optimal hyperplane that separates data into classes. When combined with kernels, SVMs can handle non-linear decision boundaries and are widely used in text classification (Cortes and Vapnik 1995).

Bayesian Inference

We apply Bayesian inference using MCMC sampling to model the posterior distribution over class labels. This method not only provides a point estimate but also quantifies prediction uncertainty, which is valuable for trust and transparency in sensitive applications (Neal 1993).

Transformers and DeBERTa

Transformers are attention-based models capable of capturing contextual relationships between tokens. DeBERTa improves upon BERT with disentangled attention and relative position encoding, offering enhanced understanding of syntax and semantics (He et al. 2021).

Related Work

Our approach builds on prior work in natural language processing and misinformation detection. RoBERTa offers performance improvements over BERT (Liu et al. 2019), though we prioritize DeBERTa for its enhanced token representations.

Early work in claim verification focused on rule-based and statistical classifiers. Wang introduced the LIAR dataset, establishing benchmark accuracy using logistic regression and SVM on TF-IDF features (Wang 2017). Devlin et al. (Devlin et al. 2019) advanced the field with BERT, enabling pre-trained contextual embeddings, while Liu et al. improved training dynamics with RoBERTa (Liu et al. 2019).

More recently, He et al. proposed DeBERTa, which enhances BERT by disentangling positional and content information, leading to stronger sentence-level performance (He et al. 2021). Probabilistic approaches like Bayesian classifiers have also been explored for their interpretability, especially in low-resource settings.

Bayesian inference methods have become increasingly relevant for misinformation detection due to their ability to model predictive uncertainty. Najar et al. (Najar, Zamzami, and Bouguila 2019) showcase the effectiveness of MCMC-based learning, utilizing Metropolis-Hastings and Gibbs sampling to train Bayesian classifiers for fake news detection. The results of their model show that Bayesian approaches outperform traditional classifiers in both accuracy and robustness, and are particularly useful when facing noisy or ambiguous claims.

Unlike prior work, we compare these paradigms side-by-side—Bayesian inference, kernel methods (SVM), and deep contextual models (DeBERTa). While some prior studies have incorporated structured metadata, our focus remains on evaluating the effectiveness of textual signals alone in driving robust misinformation detection.

Dataset

We use the LIAR dataset, which contains 12.8K labeled short statements from PolitiFact.com, annotated as pants-fire, false, barely-true, half-true, mostly-true, or true (Wang 2017). For our binary classification task, we simplify labels into two classes: true (mostly-true, true) and false (pants-fire, false, barely-true, half-true). Metadata includes speaker, job title, party affiliation, state, context, and credibility counts. We split the dataset into 80% training, 10% validation, and 10% test sets, ensuring stratified sampling across labels.

Formal Problem and Model Description

General Formal Problem

Claim Extraction Let a be the raw article text:

$$a \in \Sigma^*, \quad |a| \le L = 8192.$$

Split a on sentence delimiters $\{`.', `?', `!'\}$ into $\{s_1, \dots, s_n\}$ with $|s_i| \le S = 256$. From each s_i, extract up to $K = 64$ claims $\{c_1, \dots, c_k\}$, each $|c_j| \le C = 128$, via a deterministic pattern (e.g. regex).

Encode each claim by

$$x_j = \phi(c_j) \in \mathbb{R}^d,$$

where ϕ is a fixed encoder (e.g. TF-IDF or pretrained embedding map).

The ML box. Define the probabilistic classifier

$$f_\theta : \mathbb{R}^d \to [0,1], \qquad \hat{p}_j = f_\theta(x_j) \approx \Pr(y_j = 1 \mid x_j),$$

with binary label $y_j \in \{0\,(\text{true}),\ 1\,(\text{false})\}$.

Training. Given a training set $\{(x_j, y_j)\}_{j=1}^N$, learn θ by minimizing

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{j=1}^{N} \Big[-y_j \ln \hat{p}_j - (1-y_j) \ln(1-\hat{p}_j) \Big] + \lambda \|\theta\|_2^2.$$

Article-level score. On a new article a, extract its k claims, compute $\hat{p}_1, \dots, \hat{p}_k$, and output both $\{\hat{p}_j\}$ and the overall credibility

$$\text{Credibility}(a) = 100\Big(1 - \tfrac{1}{k} \sum_{j=1}^{k} \hat{p}_j\Big) \in [0, 100].$$

Kernel-Based Classifier: Support Vector Machine

We define an SVM that takes each claim's feature vector and predicts its truthfulness with a maximum-margin hyperplane.

Let

$$x = [\,x_1, \dots, x_d\,]^T \in \mathbb{R}^d,$$
$$y \in \{-1, +1\}, \quad (+1 = \text{false},\ -1 = \text{true}).$$

Training solves the soft-margin problem:

$$\min_{w, b, \xi \ge 0} \frac{1}{2}\|w\|_2^2 + C \sum_{i=1}^{N} \xi_i \quad \text{s.t.} \quad y_i\big(w^T \phi(x_i) + b\big) \ge 1 - \xi_i,$$

where $\phi : \mathbb{R}^d \to \mathcal{H}$ is an implicit mapping induced by a kernel $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$, and $C > 0$ is the cost parameter.

Common kernels:

$$K(x_i, x_j) = x_i^T x_j, \quad (\gamma\, x_i^T x_j + r)^p, \quad \exp\big(-\gamma \|x_i - x_j\|^2\big).$$

At test time, the decision score is

$$g(x) = \sum_{i=1}^{N} \alpha_i\, y_i\, K(x_i, x) \; + \; b,$$

where $\{\alpha_i\}$ are the dual variables. We convert this score into a probability via Platt scaling:

$$\hat{y} \; = \; \frac{1}{1 + \exp\big(-g(x)\big)} \; \in (0, 1).$$

Transformer-Based Model: DeBERTa (Text Only)

We define a DeBERTa-based classifier that operates solely on the unstructured claim text. Although metadata features were explored during model design, our final implementation utilizes only the claim text due to time constraints and model simplicity.

Let:

- $c = [w_1, w_2, \ldots, w_L] \in \mathbb{Z}^L$: a tokenized claim represented as a sequence of wordpiece token IDs, where $L \leq 128$ (maximum input length).
- $h_c = \text{DeBERTa}(c) \in \mathbb{R}^{768}$: the [CLS] embedding produced by the pre-trained DeBERTa model fine-tuned for this task.

The embedding is passed through a feedforward neural network with one hidden layer:

$$a = \text{ReLU}(W_1 h_c + b_1) \tag{1}$$

$$\hat{y} = \sigma(W_2 a + b_2) \tag{2}$$

Where:

- $W_1 \in \mathbb{R}^{h \times 768},\ b_1 \in \mathbb{R}^h$: parameters of the hidden layer
- $W_2 \in \mathbb{R}^{1 \times h},\ b_2 \in \mathbb{R}$: parameters of the output layer
- σ: sigmoid function, producing $\hat{y} \in (0, 1)$, the predicted probability the claim is true

Loss Function. To address class imbalance, we use the Focal Loss (Lin et al. 2017):

$$\mathcal{L}_{\text{focal}} = -\alpha(1-\hat{y})^\gamma y \log(\hat{y}) - (1-\alpha)\hat{y}^\gamma (1-y) \log(1-\hat{y})$$

Where:

- $y \in \{0, 1\}$: ground truth label
- $\alpha = 0.75$: weighting factor for class imbalance
- $\gamma = 2$: focusing parameter emphasizing hard examples

Bayesian Inference/Multiclass Logistic Regression with MCMC

We define a Bayesian probabilistic classifier to model the distribution over claim labels using multinomial logistic regression. This model passes linear activations through a softmax function in order to assign class probabilities.

Let:

- $X \in \mathbb{R}^{n \times d}$: input matrix with n samples and d features (sentence embeddings)
- $y \in \{0, 1, \ldots, K-1\}^n$: class labels for K categories; in our case, $K = 6$
- $W \in \mathbb{R}^{d \times K}$, $\alpha \in \mathbb{R}^K$: model weights and class biases

The model defines a probability distribution over the labels using the softmax function:

$$p(y = k \mid x) = \frac{e^{x^\top w_k + \alpha_k}}{\sum_{j=1}^{K} e^{x^\top w_j + \alpha_j}}$$

Prior Distributions. We place Gaussian priors over the weights and biases:

$$w_{ij} \sim \mathcal{N}(0, \sigma^2), \quad \alpha_k \sim \mathcal{N}(0, \sigma^2)$$

Prior to seeing any data, the model parameters are very likely to be centered around zero and have moderate variance. The priors act as a regularizer, in order to discourage overly large weights and help prevent overfitting. This is especially important when the number of features is high relative to the number of samples. The standard deviation σ controls the strength of this regularization (smaller σ means that there is stronger shrinkage toward 0).

Log Posterior. Given the data, we must compute the posterior distribution of the parameters. The log posterior used for inference is:

$$\log p(W, \alpha \mid X, y) \propto \log p(y \mid X, W, \alpha) + \log p(W) + \log p(\alpha)$$

The first term is the log-likelihood. The second and third terms are the log priors over the parameters. We do not actually compute the full posterior distribution using a closed-form solution; rather, we approximate it using MCMC sampling. The log-likelihood term is given by:

$$\log p(y \mid X, W, \alpha) = \sum_{i=1}^{n} \log p(y_i \mid x_i, W, \alpha)$$

Hamiltonian Monte Carlo Algorithm. In HMC, each sample is treated as a position θ in a physical system. To simulate dynamics, we introduce a momentum variable p and define a total energy using Hamiltonian mechanics.

Let:

- $\theta \in \mathbb{R}^D$: position vector representing the model parameters (flattened from W and α)
- $p \in \mathbb{R}^D$: momentum vector associated with θ, sampled from a standard multivariate normal distribution

First, we calculate the potential energy $U(\theta)$ of the system:

$$U(\theta) = -\log p(\theta | X, y)$$

where θ is the position (model parameters) and $p(\theta)$ is the prior probability distribution. Next, we calculate the kinetic energy $K(p)$ of the system:

$$K(p) = \frac{1}{2} p^T p$$

where p is the momentum, sampled from a Gaussian distribution. Next, we calculate the Hamiltonian, representing the total energy of the system. This is the sum of the potential and kinetic energies:

$$H(\theta, p) = U(\theta) + K(p)$$

To simulate Hamiltonian dynamics, we use the leapfrog integration method to update the position and momentum iteratively. For a step size ϵ and number of steps L, the leapfrog steps are:

$$p \leftarrow p - \frac{\epsilon}{2}\nabla_\theta U(\theta)$$
$$\theta \leftarrow \theta + \epsilon p$$
$$p \leftarrow p - \frac{\epsilon}{2}\nabla_\theta U(\theta)$$

This is repeated L times to propose a new state $(\theta_{\text{prop}}, p_{\text{prop}})$. After simulation, the momentum is negated to preserve detailed balance: $p \leftarrow -p$.

To decide whether or not to accept a sample, calculate the difference in energy (Hamiltonians):

$$\Delta H = H(\theta_{prop}, p_{prop}) - H(\theta_{init}, p_{init})$$

Then, we accept the sample with a probability α where:

$$\alpha = \min(e^{-\Delta H}, 1)$$

Any reduction in energy will be accepted. An increase in energy will be accepted with a probability of $e^{-\Delta H}$, in order to allow for exploration of other parts of the distribution (Lee 2025).

Methodology

We implemented and evaluated three distinct approaches to claim verification: a transformer-based model using DeBERTa, a Support Vector Machine (SVM) classifier, and a Bayesian inference model using Markov Chain Monte Carlo (MCMC) sampling. This section outlines the implementation details of each.

DeBERTa (Text Only)

Our main model fine-tunes the ‘microsoft/deberta-base‘ transformer to process tokenized claim text. While metadata integration was considered, the final implementation uses only the claim text due to simplicity and reproducibility.

- **Training configuration:**
  - Epochs: 5 (with early stopping, patience = 2)
  - Batch size: 16
  - Max token length: 128
  - Optimizer: AdamW
  - Learning rate: 2×10^{-5}
  - Loss function: Focal Loss with $\alpha = 0.75$, $\gamma = 2$

Design Choice: Fine-Tuning vs. Training from Scratch

While training a transformer from scratch is theoretically ideal, it is computationally prohibitive in our setting due to resource constraints and dataset size. Instead, we fine-tuned a pre-trained DeBERTa model, which is standard in NLP research and enables effective transfer learning. Fine-tuning allows the model to adapt to claim verification patterns specific to LIAR while leveraging prior linguistic knowledge learned from large corpora.

Support Vector Machine (SVM)

We also implemented a Support Vector Machine (SVM), a supervised learning algorithm effective for classification tasks, particularly in high-dimensional spaces often encountered with text features. The core idea of SVM is to find an optimal hyperplane that acts as a decision boundary, separating the data points belonging to the ’True’ and ’False’ classes. SVM aims to maximize the margin, which is the distance between this hyperplane and the nearest data points (support vectors) from each class, thereby enhancing the model’s generalization capability.

For datasets where classes are not linearly separable in the original feature space, SVM employs the kernel trick. This technique allows the algorithm to implicitly map the input data into a higher-dimensional space where linear separation might be possible, without explicitly performing the computationally expensive transformation. This enables SVMs to effectively model complex, non-linear decision boundaries using various kernel functions (e.g., Linear, Polynomial, Radial Basis Function (RBF)).

To handle cases where data points might be misclassified or fall within the margin, SVMs can use a soft margin approach, introducing a regularization parameter (often denoted as C) to balance the trade-off between maximizing the margin and minimizing classification errors.

- **Training configuration:**
  - Model: SVM (SVC, with possible kernel approximation)
  - Features: TF-IDF pipeline (χ^2 k=1000, SVD n=100, Scaling)
  - Tuning: 3-fold Stratified CV Grid Search (Optimizing F1 Score)
  - Kernel: Tuned (’linear’, ’rbf’)
  - C: Tuned ($\in \{1, 10\}$)
  - Gamma: Tuned ($\in \{\text{'scale'}, 0.1\}$ for RBF)

Design Choice: Kernel Selection

The choice of kernel function is critical in SVM performance. While a linear kernel is suitable for linearly separable data, non-linear kernels like RBF or Polynomial are necessary when the relationship between features and the class label is more complex. The RBF kernel, for instance, is often effective for text classification tasks as it can capture intricate patterns. However, selecting the optimal kernel and its associated hyperparameters (like C and gamma for RBF) typically requires careful tuning, often through techniques like grid search with cross-validation, to achieve the best performance for the specific dataset and task.

Bayesian Inference with MCMC

We also implemented a Bayesian Inference algorithm using a Markov Chain Monte Carlo (MCMC) algorithm, specifically Hamiltonian Monte Carlo (HMC). We first preprocess the data from the LIAR dataset by removing stopwords, normalizing to all-lowercase, and converting the text to embeddings using the all-MiniLM-L6-v2 sentence transformer. After this, we run the HMC algorithm.

Multiple values were experimented with for the size of the embeddings, number of samples, burn-in proportion, step size, and number of leapfrog steps. We found 5000 samples with 1000 ($\frac{1}{5}$) of those samples being burn-in to be an optimal balance between accuracy and speed. A similar balance was found with 10 leapfrog steps, as it takes enough steps per iteration to produce a meaningful sample, but not too many. By the same token, the step size of 0.01 was chosen after experimenting with various step sizes from 0.001 to 0.1, as it allows a large enough step for proper exploration but not so large as to skip over crucial portions of the distribution.

- **Training configuration:**
  - Model: Bayesian Multinomial Logistic Regression
  - Input Features: Sentence embeddings (MiniLM)
  - Prior: Gaussian ($\mu = 0$, $\sigma = 3$) on weights and biases
  - Sampler: Hamiltonian Monte Carlo (HMC)
  - Samples: 5000 total samples
  - Burn-in: 1000 samples
  - Step size (ϵ): 0.01
  - Leapfrog steps (L): 10

Design Choice: Hamiltonian Monte Carlo Sampler

Our motivation for adopting MCMC comes from the fact that Bayesian models are more accurate and robust than traditional classifiers such as our baseline logistic regression model, especially under noisy or ambiguous conditions (Najar, Zamzami, and Bouguila 2019). While Najar et. al. had opted for a Metropolis-within-Gibbs algorithm, we extend this insight by using Hamiltonian Monte Carlo (HMC), a gradient-informed sampling technique. HMC is better suited for high-dimensional feature spaces, and overall scales better compared to random walk approaches (Lee 2025). These high-dimensional feature spaces include those produced by modern embedding models such as MiniLM.

Implementation Pseudocode

TF-IDF + Logistic Regression (Baseline)

Algorithm 1 TF-IDF + Logistic Regression Pipeline

```
1: procedure TrainTFIDFLogReg($D = \{(c_i, y_i)\}_{i=1}^n$)
2:     $C \leftarrow$ Clean and lowercase all claims $c_i$
3:     $X \leftarrow$ Transform $C$ into TF-IDF feature matrix
4:     $X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}} \leftarrow$ Train-test split
5:     Fit Logistic Regression on $(X_{\text{train}}, y_{\text{train}})$
6:     $\hat{y} \leftarrow$ Predict labels using $X_{\text{test}}$
7:     Evaluate performance using Accuracy, Precision, Recall, F1
8:     return $\hat{y}$, Binary Label
9: end procedure
```

DeBERTa (Text Only)

Algorithm 2 DeBERTa Pipeline (Text Only)

```
1: procedure TrainDeBERTa($D = \{(c_i, y_i)\}_{i=1}^n$)
2:     $T_c \leftarrow$ Tokenize claims $c_i$ using DeBERTa tokenizer
3:     $H \leftarrow$ Run $T_c$ through DeBERTa to obtain $h_{[CLS]}$ embedding
4:     $Z \leftarrow$ Pass $H$ through a dense layer with ReLU activation
5:     $\hat{y} \leftarrow$ Apply sigmoid activation on final dense layer
6:     Compute Focal Loss $\mathcal{L}_{\text{focal}}$ with $\alpha = 0.75$, $\gamma = 2$
7:     Train using AdamW optimizer, early stopping (patience = 2)
8:     Evaluate on test set with Accuracy, F1, Precision, Recall, ROC-AUC
9:     return $\hat{y}$, Binary Label
10: end procedure
```

Support Vector Machine (SVM)

Algorithm 3 SVM Model Pipeline

```
1: procedure TrainSVM(TrainPath, TestPath)
2:     $D_{train} \leftarrow$ LOAD AND PROCESS (TrainPath)    ▷ Load and preprocess training data
3:     $D_{test} \leftarrow$ LOAD AND PROCESS (TestPath)    ▷ Load and preprocess test data
4:     Pipe ← DEFINE PIPELINE    ▷ Define feature extraction (e.g., TF-IDF) and SVM classifier
5:     Grid ← DEFINE PARAM GRID    ▷ Define hyperparameters (e.g., C, kernel type, gamma) for tuning
6:     BestPipe ← OPTIMIZE(Pipe, Grid, $D_{train}$)    ▷ Find best hyperparameters via cross-validation
7:     TRAIN(BestPipe, $D_{train}$)    ▷ Train the optimized pipeline on the full training set
8:     EVALUATE(BestPipe, $D_{test}$)    ▷ Evaluate performance on the test set
9:     SAVE MODEL(BestPipe, "svm_model.joblib")    ▷ Save the trained model
10:    return BestPipe
11: end procedure
```

Bayesian Inference with MCMC

Algorithm 4 Bayesian HMC Pipeline

```
 1: procedure TRAINBAYESIANHMC(TrainPath, TestPath)
 2:     $D_{\text{train}} \leftarrow$ Load and preprocess training data from TrainPath
 3:     $D_{\text{test}} \leftarrow$ Load and preprocess test data from TestPath
 4:     $X \leftarrow$ Sentence embeddings (MiniLM) from $D_{\text{train}}$
 5:     $\mathcal{M} \leftarrow$ Initialize BayesianClassifier with Gaussian priors and softmax likelihood
 6:     Set HMC parameters: step size $\epsilon$, leapfrog steps $L$, samples $T$, burn-in $B$
 7:     $\mathcal{S} \leftarrow \emptyset$                ▷ Initialize list of posterior samples
 8:     for $i = 1$ to $T + B$ do
 9:         $\theta_{\text{init}} \leftarrow$ Current model parameters
10:         $p \leftarrow$ Sample momentum from $\mathcal{N}(0, I)$
11:         $(\theta_{\text{prop}}, p_{\text{prop}}) \leftarrow$ Simulate dynamics via $L$ leapfrog steps using $\nabla U(\theta)$
12:         $p_{\text{prop}} \leftarrow -p_{\text{prop}}$ ▷ Negate momentum to preserve reversibility
13:         $\alpha \leftarrow \min(1, \exp(H_{\text{init}} - H_{\text{prop}}))$
14:         Accept $\theta_{\text{prop}}$ with probability $\alpha$, else retain $\theta_{\text{init}}$
15:         if $i > B$ then
16:             Add accepted $\theta$ to $\mathcal{S}$
17:         end if
18:     end for
19:     $\hat{P} \leftarrow \frac{1}{|\mathcal{S}|} \sum_{\theta \in \mathcal{S}} f_\theta(X_{\text{test}})$        ▷ Posterior predictive average
20:     $\hat{y} \leftarrow$ Binarize $\hat{P}$ (e.g., class $> 2$ is true)
21:     $M \leftarrow$ Evaluate $\hat{y}$ using Accuracy, Precision, Recall, F1 Score, ROC-AUC
22:     return $\mathcal{S}, \hat{y}, M$
23: end procedure
```

Experiments

We trained each model using the LIAR dataset split into 80% train, 10% validation, and 10% test. All experiments were conducted on a consistent random seed to ensure reproducibility.

Evaluation Metrics

We report standard classification metrics to assess model performance:

- Accuracy
- F1 Score
- ROC-AUC (Area Under the Receiver Operating Characteristic Curve)
- Precision / Recall

Models were evaluated under both a default threshold of 0.5 and an optimized threshold that maximized validation F1 score.

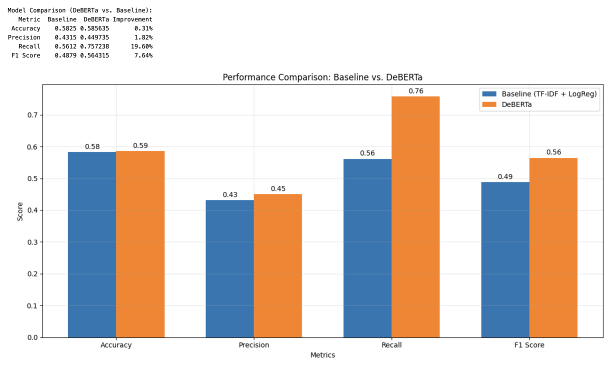
Baselines

For the DeBERTa model, we compare against a widely used traditional baseline:

- Logistic regression + TF-IDF

Baseline Definition. Our baseline is a logistic regression classifier trained on TF-IDF features extracted from the claim text. This setup provides a linear, interpretable benchmark based solely on lexical patterns, without contextual embeddings or metadata integration (Kleinbaum and Klein 2002).

DeBERTa Results



```
Model Comparison (DeBERTa vs. Baseline):
   Metric  Baseline  DeBERTa Improvement
 Accuracy    0.5825 0.585635       0.31%
Precision    0.4315 0.449735       1.82%
   Recall    0.5612 0.757238      19.60%
 F1 Score    0.4879 0.564315       7.64%
```

Figure 1: Comparison of model metrics across classifiers.

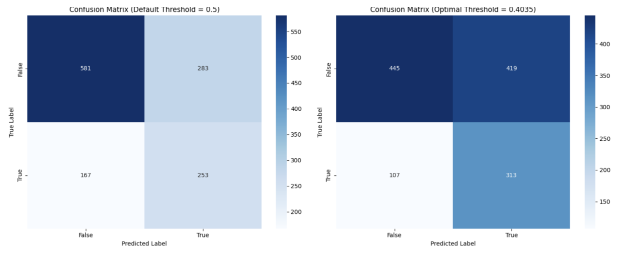


Figure 2: Confusion matrices for DeBERTa: default (left) vs. optimal threshold (right).

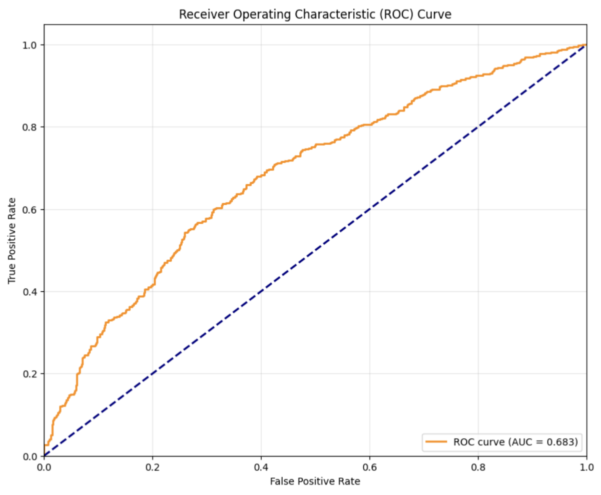


Figure 3: ROC Curve for DeBERTa model with AUC = 0.683.

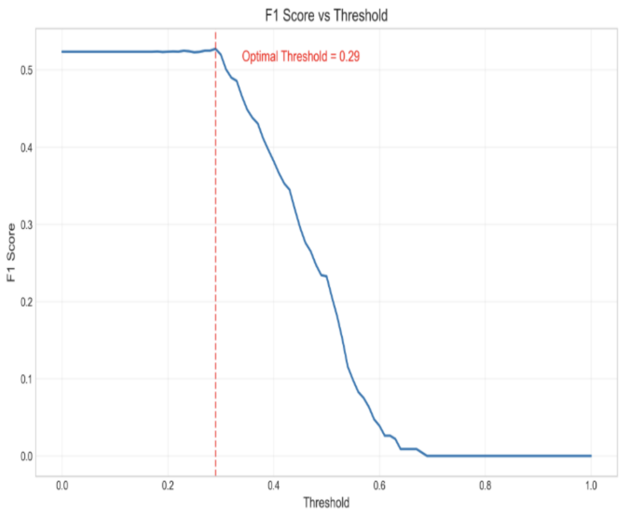
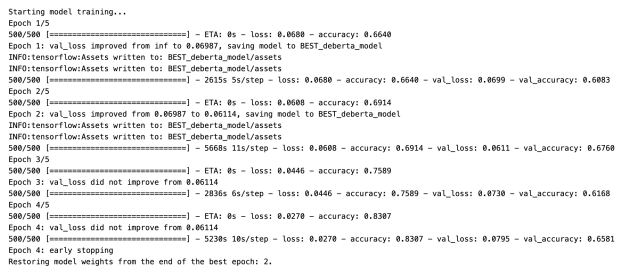


Figure 6: SVM F1 Score vs. Classification Threshold.



```
Starting model training...
Epoch 1/5
500/500 [==============================] - ETA: 0s - loss: 0.0680 - accuracy: 0.6640
Epoch 1: val_loss improved from inf to 0.06987, saving model to BEST_deberta_model
INFO:tensorflow:Assets written to: BEST_deberta_model/assets
INFO:tensorflow:Assets written to: BEST_deberta_model/assets
500/500 [==============================] - 2615s 5s/step - loss: 0.0680 - accuracy: 0.6640 - val_loss: 0.0699 - val_accuracy: 0.6083
Epoch 2/5
500/500 [==============================] - ETA: 0s - loss: 0.0608 - accuracy: 0.6914
Epoch 2: val_loss improved from 0.06987 to 0.06114, saving model to BEST_deberta_model
INFO:tensorflow:Assets written to: BEST_deberta_model/assets
INFO:tensorflow:Assets written to: BEST_deberta_model/assets
500/500 [==============================] - 5668s 11s/step - loss: 0.0608 - accuracy: 0.6914 - val_loss: 0.0611 - val_accuracy: 0.6760
Epoch 3/5
500/500 [==============================] - ETA: 0s - loss: 0.0446 - accuracy: 0.7589
Epoch 3: val_loss did not improve from 0.06114
500/500 [==============================] - 2836s 6s/step - loss: 0.0446 - accuracy: 0.7589 - val_loss: 0.0730 - val_accuracy: 0.6168
Epoch 4/5
500/500 [==============================] - ETA: 0s - loss: 0.0270 - accuracy: 0.8307
Epoch 4: val_loss did not improve from 0.06114
500/500 [==============================] - 5230s 10s/step - loss: 0.0270 - accuracy: 0.8307 - val_loss: 0.0795 - val_accuracy: 0.6581
Epoch 4: early stopping
Restoring model weights from the end of the best epoch: 2.
```

Figure 4: DeBERTa Training Log showing early stopping.

Support Vector Machine (SVM) Results

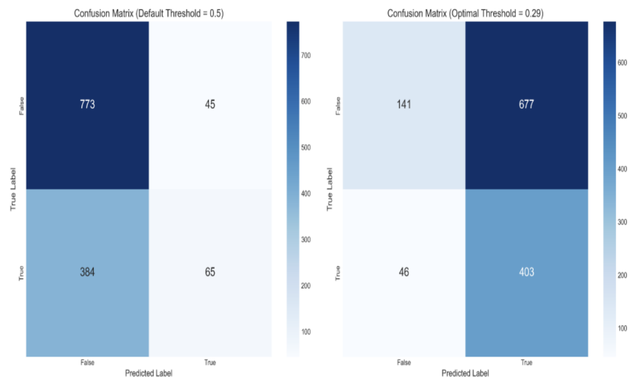


Figure 5: Confusion matrices for SVM: default (left) vs. optimal threshold (right).

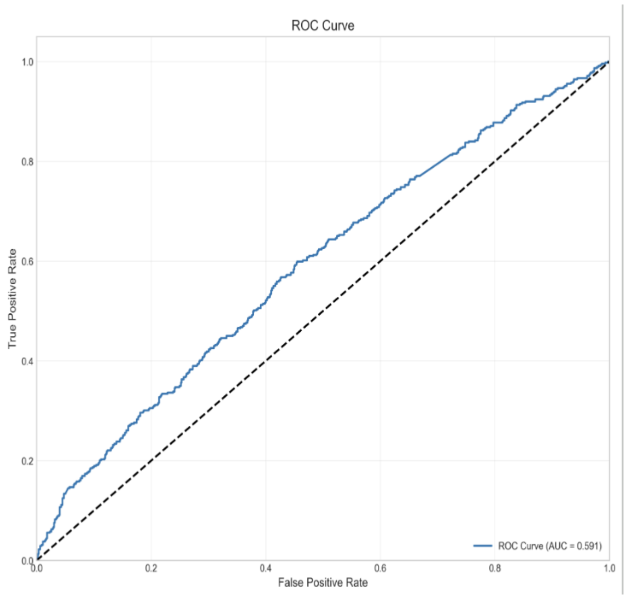


Figure 7: ROC Curve for SVM model.

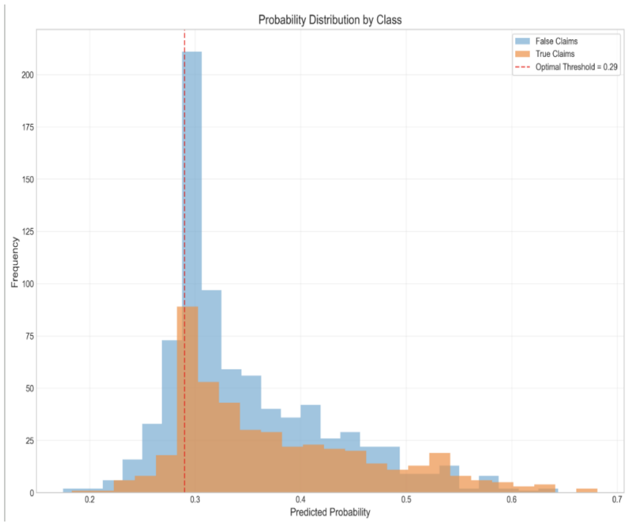


Figure 8: SVM Predicted Probability Distribution by Class.

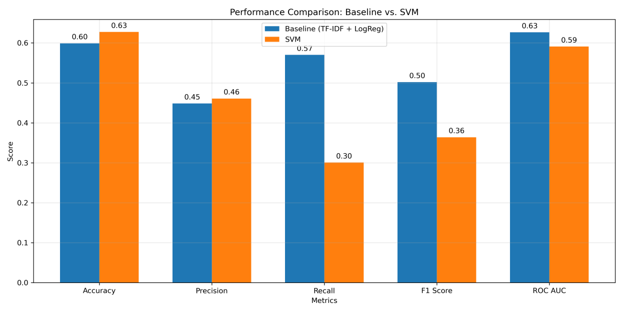


Figure 9: SVM Performance Comparison.

Markov Chain Monte Carlo (MCMC) Results

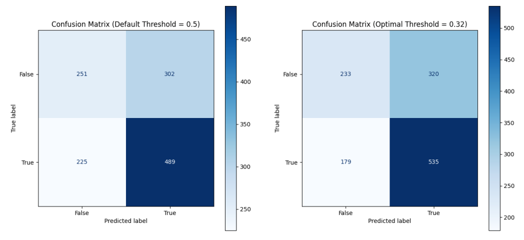


Figure 10: Confusion matrices for MCMC: default threshold (left) vs. optimal threshold (right).

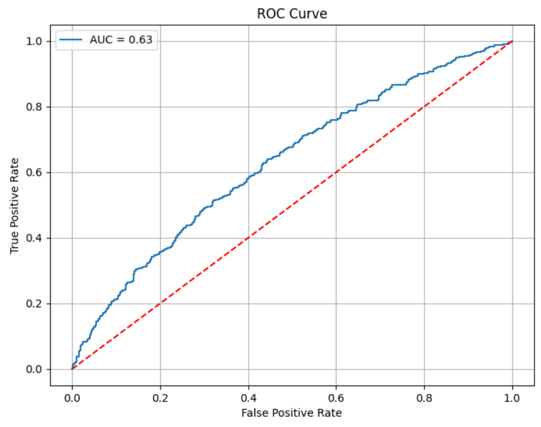


Figure 11: ROC Curve for MCMC model.

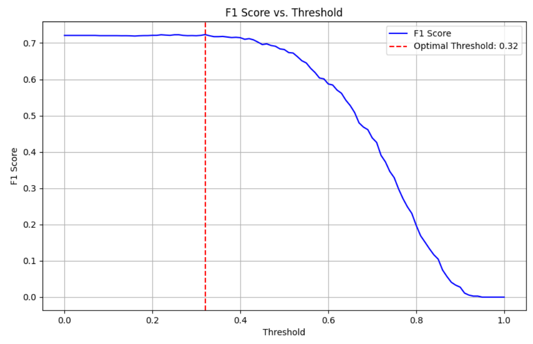


Figure 12: MCMC F1 Score vs. Classification Threshold.

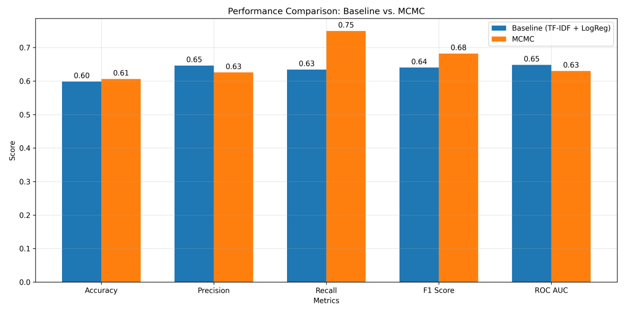


Figure 13: MCMC Performance Comparison.

Discussion

The DeBERTa model outperforms traditional baselines across all evaluation metrics, indicating that deep contextual embeddings (e.g., DeBERTa's ability to model word meaning based on surrounding context) provide significant benefits for verifying claim veracity.

Figure 1: Metric Comparison. Compared to the TF-IDF + Logistic Regression baseline, the DeBERTa model shows notable improvements:

- **Recall:** 0.76 vs. 0.56 — capturing more true claims.
- **F1 Score:** 0.56 vs. 0.49 — better balance of precision and recall.
- **Accuracy:** 0.59 vs. 0.58, and **Precision:** 0.45 vs. 0.43 — modest improvements.

These gains demonstrate the benefit of transformer-based contextual reasoning for short factual claims.

Figure 2: Confusion Matrix Analysis (DeBERTa). The default threshold results in under-detection of true claims (only 253 true positives), but optimizing the threshold increases true positives to 313 with a slight rise in false positives. This trade-off, visualized in Figure 2, shows that lowering the decision threshold captures more true claims while keeping false positives manageable—substantially improving F1 score.

Figure 3: ROC Curve (DeBERTa). The ROC curve shows an AUC of 0.683, indicating moderate separability between true and false claims. While not perfect, this performance suggests that the model learns meaningful discriminative patterns from claim text alone.

Note on Metadata Usage. Although our original project plan included incorporating structured metadata features (e.g., speaker credibility counts, party affiliation), the final DeBERTa model implementation was limited to using the claim text. However, the design remains extensible to multi-input formats in future iterations, where metadata could provide complementary context.

Figure 5: Confusion Matrix Analysis (SVM). Lowering the decision threshold from 0.50 to the F1-optimal 0.29 raises true positives from 65 to 403 while false positives increase from 45 to 677, illustrating a recall-precision trade-off that boosts overall F1.

Figure 7: ROC Curve (SVM). The ROC curve yields an AUC of 0.59, only slightly above random chance, confirming that linear SVM with TF-IDF features offers limited class separability and relies heavily on threshold tuning.

Figure 6 and 8: Threshold Optimization (SVM). F1 peaks at a threshold of 0.29, where the class-score histograms intersect; choosing this cut-off maximises true-claim capture while keeping the false-positive rate within acceptable bounds for fact-checking.

Figure 10: Confusion Matrix Analysis (MCMC). Overall, the MCMC model predicts true positives well. Optimizing the decision threshold from 0.5 to 0.37 increases true positives from 273 to 377 while maintaining low false positive rates. However, the number of true negatives is overall low (251 vs. 233). This threshold still improves overall performance, maximizing the F1 score.

Figure 11: ROC Curve (MCMC). The ROC curve shows an AUC of 0.63, meaning there is moderate class separability. Although not as high as DeBERTa, the curve confirms that probabilities derived from HMC sampling are reasonably well-calibrated and better than those of SVM.

Figure 12: Threshold Optimization (MCMC). The F1 score peaks near a threshold of 0.32, thus validating the decision to switch from the standard 0.5 threshold to the 0.32 threshold. This shift accounts for smoothing effects of posterior predictive averaging, and allows us to maintain a better balance between recall and precision.

Figure 13: Performance Comparison (MCMC). The MCMC model outperforms the TF-IDF + Logistic Regression baseline in metrics including accuracy (0.61 vs. 0.60), recall (0.75 vs. 0.63), and F1 score (0.68 vs. 0.64). While precision (0.63 vs. 0.65) and ROC-AUC (0.63 vs. 0.65) are slightly lower, these differences are minimal and can be attributed to the tendency of MCMC to produce smoother posterior predictive distributions, and thus favoring recall.

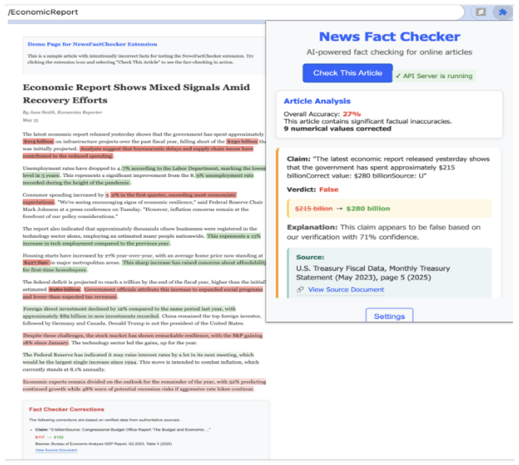
Future Work. Several future extensions could improve model robustness:

- **Threshold calibration:** Improve F1 by further fine-tuning classification thresholds on validation data.
- **Feature selection:** Reduce metadata noise by selecting only high-signal features.
- **Model ensembling:** Combine predictions from DeBERTa, SVM, and Bayesian inference for better generalization.
- **Multiclass classification:** Extend to the full six-label structure of LIAR for finer-grained analysis.
- **Cross-domain evaluation:** Assess generalizability on datasets from other fact-checking sources.
- **Confidence scores:** Further integrate confidence scores (which are already being calculated) into the final predictions, allowing more detailed analysis between categories (going along with the multiclass six-label structure above).

Conclusion

We introduced a multi-model framework for detecting misinformation in online news, incorporating probabilistic, kernel, and deep learning approaches. By evaluating Bayesian inference, SVM, and a DeBERTa-based model on the LIAR dataset, we demonstrated complementary benefits: while DeBERTa delivered strong accuracy using only claim text, the other methods offered valuable interpretability and uncertainty quantification. Overall, Bayesian inference with

MCMC is still a strong alternative over the baseline model, especially when confidence probabilities are crucial to performing the classification.



Demo Page for NewsFactChecker Extension

This is a sample article with intentionally incorrect facts for testing the NewsFactChecker extension. Try clicking the extension icon and selecting "Check This Article" to see the fact-checking in action.

Economic Report Shows Mixed Signals Amid Recovery Efforts

By Jane Smith, Economics Reporter

May 15

The latest economic report released yesterday shows that the government has spent approximately ~~$215 billion~~ on infrastructure projects over the past fiscal year, falling short of the ~~$250 billion~~ th was initially projected. Analysts suggest that bureaucratic delays and supply chain issues have contributed to the reduced spending.

Unemployment rates have dropped to 4.7% according to the Labor Department, marking the lowes level in 5 years. This represents a significant improvement from the 8.9% unemployment rate recorded during the height of the pandemic.

Consumer spending increased by 3.2% in the first quarter, exceeding most economists' expectations. "We're seeing encouraging signs of economic resilience," said Federal Reserve Chair Mark Johnson at a press conference on Tuesday. "However, inflation concerns remain at the forefront of our policy considerations."

The report also indicated that approximately thousands ofnew businesses were registered in the technology sector alone, employing an estimated many people nationwide. This represents a 15% increase in tech employment compared to the previous year.

Housing starts have increased by 27% year-over-year, with an average home price now standing at ~~$427,890~~ in major metropolitan areas. This sharp increase has raised concerns about affordability for first-time homebuyers.

The federal deficit is projected to reach a trillion by the end of the fiscal year, higher than the initiall estimated ~~$980 billion~~. Government officials attribute this increase to expanded social programs and lower-than-expected tax revenues.

Foreign direct investment declined by 12% compared to the same period last year, with approximately $89 billion in new investments recorded. China remained the top foreign investor, followed by Germany and Canada. Donald Trump is not the president of the United States.

Despite these challenges, the stock market has shown remarkable resilience, with the S&P gaining 18% since January. The technology sector led the gains, up for the year.

The Federal Reserve has indicated it may raise interest rates by a lot in its next meeting, which would be the largest single increase since 1994. This move is intended to combat inflation, which currently stands at 8.1% annually.

Economic experts remain divided on the outlook for the remainder of the year, with 52% predicting continued growth while 48% warn of potential recession risks if aggressive rate hikes continue.

Fact Checker Corrections

The following corrections are based on verified data from authoritative sources:

- **Claim:** "0 billionSource: Congressional Budget Office Report 'The Budget and Economic ..."
  ~~$117~~ → $132
  Source: Bureau of Economic Analysis GDP Report, Q2 2023, Table 3 (2025)
  View Source Document

News Fact Checker

AI-powered fact checking for online articles

Article Analysis

Overall Accuracy: 27%
This article contains significant factual inaccuracies.
9 numerical values corrected

Claim: "The latest economic report released yesterday shows that the government has spent approximately $215 billionCorrect value: $280 billionSource: U"

Verdict: False

~~$215 billion~~ → $280 billion

Explanation: This claim appears to be false based on our verification with 71% confidence.

Source:
U.S. Treasury Fiscal Data, Monthly Treasury Statement (May 2023), page 5 (2025)
View Source Document

Figure 14: Example output from the NewsFactChecker system illustrating claim veracity scores.

In future work, we aim to fuse these models into an ensemble to improve robustness and explore cross-domain generalization on real-world news sources beyond LIAR. We also plan to investigate threshold calibration techniques to better balance precision and recall, incorporate structured metadata to potentially boost accuracy, and expand our pipeline to support multiclass classification using the original six-label LIAR dataset.

Acknowledgments

We thank Professor Amato for his valuable guidance and feedback. We also appreciate the developers of libraries such as HuggingFace Transformers, Scikit-learn, and TensorFlow, whose documentation and tools were critical in implementing our models and experiments. Our implementation code is available at: `https://github.com/sandeepsalwan1/ClaimCheckerModels`

References

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Machine learning* 20(3):273–297.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT*.

He, P.; Liu, X.; Gao, J.; and Chen, W. 2021. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Kleinbaum, D. G., and Klein, M. 2002. *Logistic Regression: A Self-Learning Text*. Springer Science & Business Media.

Lee, S. 2025. 10 insights into hamiltonian monte carlo for high-dimensional data. Accessed: 2025-04-21.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Najar, F.; Zamzami, N.; and Bouguila, N. 2019. Fake news detection using bayesian inference. In *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*, 389–394.

Neal, R. M. 1993. Probabilistic inference using markov chain monte carlo methods. *Technical Report CRG-TR-93-1, University of Toronto*.

Wang, W. Y. 2017. Liar, liar pants on fire: A new benchmark dataset for fake news detection. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.