Automatic alert generation with NER and SA

Ulises Díez Santaolalla
Mathematical Engineering and AI
Pontifical University Comillas
ulises.diez@alu.comillas.edu

Teresa Franco Corzo
Mathematical Engineering and AI
Pontifical University Comillas
teresa.franco@alu.comillas.edu

Ignacio Felices Vera
Mathematical Engineering and AI
Pontifical University Comillas
ignacio.felices@alu.comillas.edu

Maria Ascanio Morcillo
Mathematical Engineering and AI
Pontifical University Comillas
maria.ascanio@alu.comillas.edu

Abstract

This project presents an automatic alert generation system based on Named Entity Recognition (NER) and Sentiment Analysis (SA), applied to textual inputs from news articles and social media posts. The system identifies key entities (such as people, organizations, or locations) and evaluates the sentiment surrounding them to generate concise, context-aware alerts. The architecture includes custom-trained neural networks for both NER and SA, and combines their outputs to produce alert sequence-to-sequence neural generation using a pre-trained model. Our aim is to combine a pipeline that includes the basic process, customized alerts and multi modal inputs, including the process of images through image captioning pre-trained models. This approach enables efficient news monitoring and early detection of relevant social, economic, or geopolitical alerts. The full architecture is illustrated in Figure 7. All the code referenced in this project can be found in the GitHub repository.[1]

1 Related Work

The task of automatic alert generation from textual sources integrates several areas from Natural Language Processing (NLP), particularly *Named Entity Recognition* (NER), *Sentiment Analysis* (SA), sequence-based text generation, and multi modal modeling. In this section, we review significant previous contributions that serve as the foundation for our system.

1.1 Sentiment Analysis

> "Sentiment analysis is the process of determining the sentiment polarity (positivity, neutrality, or negativity) of text" [1, p. 1].

Deep learning techniques have significantly advanced the field of sentiment analysis, as summarized in the survey by Zhang et al. [2]. They categorize neural approaches into convolutional, recurrent, and recursive models, each with trade-offs in interpretability and performance. Moreover, Zhou et al. [3] introduce an attention-based BiLSTM model, demonstrating how attention mechanisms can further enhance the interpretability and performance of sequence models in capturing salient features for classification.

[1] https://github.com/Ulisesdz/-Automatic-alert-generation-with-NER-and-SA.git

1.2 Named Entity Recognition

Neural architectures, particularly the BiLSTM-CRF model introduced by Lample et al., demonstrated state-of-the-art performance across multiple languages without the need for manual feature engineering [4]. More recently, Yadav and Bethard [5] provided a comprehensive survey of deep neural NER systems, emphasizing the benefits of minimal feature engineering and the role of modern context.

1.3 Text Generation and Attention Mechanisms

The Transformer architecture by Vaswani et al. [6] fundamentally changed sequence modeling by introducing self-attention as an alternative to recurrence, enabling better long-range dependency modeling and parallelization. These innovations inspired a new generation of encoder-decoder models used in both translation and generative tasks.

For multi modal generation tasks, works such as *Show and Tell* [7] and *Show, Attend and Tell* [8] explored image captioning with deep convolutional encoders and attention-based LSTM decoders.

2 Models Design

2.1 Sentiment Analysis Model

This section outlines the development of a Sentiment Analysis (SA) model, leveraging the vast amount of informal, user-generated content shared daily on social media. The model was trained on the *Sentiment140* dataset, which contains 1.6 million tweets labeled as positive (4) or negative (0). This dataset, widely used in academic research, offers a reliable foundation for analyzing informal texts, as it includes typical social media elements such as abbreviations, emoticons, hashtags, and slang [9].

2.1.1 Model Development

The proposed model for Sentiment Analysis is a Recurrent Neural Network (RNN), specifically using Long Short-Term Memory (LSTM) due to its ability to model long-term dependencies, which is crucial in sentiment analysis where key words at the end of a sentence can alter its meaning. As noted by Shobana and Murali [1, p. 1], deep neural networks have significantly improved performance in various NLP tasks, including sentiment analysis. LSTM, a special variation of RNN, has shown to excel in sentiment analysis [10], capturing sentiment information more effectively and improving classification accuracy.

In our experimental evaluations, we found that a certain level of architectural complexity does not necessarily hurt performance if regularization is properly managed. The architecture consists of pretrained word embeddings followed by a three-layer bidirectional LSTM with 128 units per direction. An attention mechanism was integrated to help the model focus on relevant parts of the sequence before classification.

This design was inspired by the "Bidirectional LSTM model with self-attention mechanism and multi-channel features (SAMF-BiLSTM)" [11], though we omitted the multi-channel component in favor of using only pretrained embeddings. It also draws from the idea in "BiLSTM-SNP to capture semantic correlations" [12], replacing the Spiking Neural P System mechanism with standard activation functions.

2.1.2 Model Architecture and Training Strategy

The resulting architecture, grounded in the aforementioned ideas, is illustrated in Figure 1 [3, p. 209].

To stabilize training and improve generalization, Layer Normalization was applied before the LSTM layer. This choice is supported by research showing that "LSTM with layer normalization gives slightly better results compared to LSTM" [13, p. 1]. Unlike Batch Normalization, which requires separate statistics for each time step and complicates generalization to longer sequences, Layer Normalization normalizes based only on activations within each time step [14].

The model is trained with a batch size of 64, a learning rate of 0.001, and a weight decay of 5e-4 as additional regularization. A 30% dropout rate is applied within the LSTM to reduce overfitting.

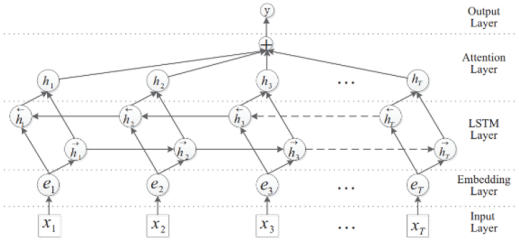

Figure 1: Bidirectional LSTM model with attention.

For optimization, the Adam algorithm is used due to its efficiency, stability, and adaptability in NLP tasks. Adam dynamically adjusts the learning rate for each parameter by combining the advantages of algorithms like AdaGrad and RMSProp [15].

The `BCEWithLogitsLoss` function was selected for the sentiment analysis model due to the binary nature of the classification task. The dataset contains only positive and negative samples, encoded as classes 0 and 2. During training, labels with value 2 are mapped to 1 to align with the binary loss function. This transformation does not affect inference, where the original 0/2 encoding is preserved, allowing future expansion to ternary classification.

This loss function is advantageous as it incorporates the sigmoid activation internally, reducing the need for manual application and minimizing numerical instability during training.

Since the model was trained only on positive (2) and negative (0) classes, a double-threshold post-processing strategy was applied during inference to introduce a neutral class (1):

- Probability < 0.35: label 0 (negative)
- Probability between 0.35 and 0.65: label 1 (neutral)
- Probability > 0.65: label 2 (positive)

This strategy is inspired by three-way decision theory [16], which categorizes predictions into positive, negative, and boundary regions to better handle uncertainty.

2.2 Named Entity Recognition Model

This section outlines the design and implementation of the Named Entity Recognition (NER) model, which adopts the BIO tagging scheme. In this framework, each token is labeled as the beginning (B), inside (I), or outside (O) of a named entity span. For example, the tag `B-PER` denotes the beginning of a person entity, while `I-PER` indicates a subsequent token within the same entity.

The model was trained on the widely-used *CoNLL-2003* dataset, a benchmark corpus consisting of annotated sentences from newswire articles. This dataset provides tokenized words alongside their respective NER tags, facilitating robust model training and enabling meaningful comparison with existing approaches in the literature.

2.2.1 Model Development

To effectively capture contextual dependencies critical in NER tasks, a Bidirectional Long Short-Term Memory (BiLSTM) architecture was adopted from [4] . As shown in Figure 2, the model comprises four principal layers: a trainable word embedding layer, a bidirectional LSTM layer, and a linear classification layer. This structure enables the model to learn representations based on both preceding and succeeding tokens, which is essential for accurately identifying entity boundaries and categories.

The use of BiLSTM is motivated by the nature of NER, where the classification of a word often depends heavily on its surrounding context. The bidirectional structure ensures that both left and right contexts contribute to the representation of each token. This design draws inspiration from the architecture proposed in “Neural Architectures for Named Entity Recognition”, with several modifications tailored to this implementation. Instead of using pretrained embeddings such as Word2Vec, the model uses trainable embeddings, allowing the representation space to adapt directly

to the CoNLL-2003 dataset. Furthermore, batch normalization is applied to enhance training stability and reduce internal covariate shift.

2.2.2 Model Architecture and Training Strategy

The final architecture, based on the aforementioned principles, is visualized in Figure 2 [17].

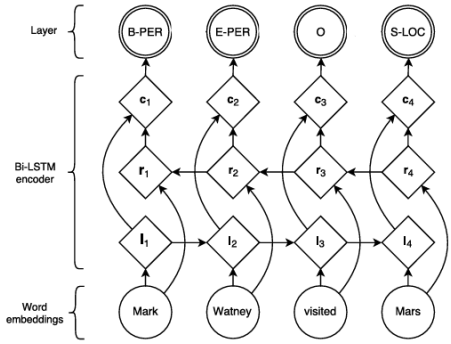

Figure 2: BiLSTM architecture for Named Entity Recognition.

The model takes a sentence $(x_1, x_2, \ldots, x_n)$ consisting of n words, where each word x_t is represented by a d-dimensional vector, a Bidirectional Long Short-Term Memory (BiLSTM) network captures contextual information from both directions of the sequence.

A forward LSTM processes the sentence from left to right, generating a sequence of hidden states $\overrightarrow{h_t}$, which encode the left-side context up to position t. Simultaneously, a backward LSTM processes the same sentence in reverse (from right to left), producing hidden states $\overleftarrow{h_t}$ that encode the right-side context.

At each time step t, the final hidden state h_t is obtained by concatenating the forward and backward representations:

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$$

This combined representation allows the model to leverage information from both preceding and succeeding tokens, which is essential in tasks like Named Entity Recognition, where the classification of a token often depends on its surrounding context [17].

Each token in the sentence is first mapped to a 300-dimensional trainable embedding. These embeddings are then passed through two stacked layers of BiLSTM with 128 hidden units per direction, forming a comprehensive 256-dimensional contextual representation per token.

Batch normalization is applied to the outputs of the BiLSTM layers to standardize feature distributions and improve generalization. Although Layer Normalization is often preferred in sequence modeling due to its per-time-step independence [14], batch normalization was found empirically to yield better performance for this NER task by leveraging normalization across batch features.

A dropout rate of 50% is introduced after the normalization layer to prevent overfitting, a common issue in deep recurrent networks. This aggressive regularization encourages robust feature learning and enhances generalization to unseen data.

The model is trained using the Adam optimizer [15], with a batch size of 32, a learning rate of 0.001, and a weight decay coefficient to encourage sparsity in the learned weights. Adam was chosen for its stability and dynamic learning rate adjustment, which are particularly beneficial in NLP tasks with high-dimensional parameter spaces. [15]

The loss function utilized is `CrossEntropyLoss`, suitable for multi-class classification problems such as NER, where each token must be assigned one of many possible tags. To mitigate the impact of class imbalance—particularly the dominance of the "O" (non-entity) tag—class weights were computed and incorporated into the loss calculation. Padding indices were excluded from the loss to ensure that only meaningful token positions contribute to parameter updates during backpropagation.

2.3 Alert Generation Model

In this work, we explore multiple approaches to automated alert generation using NLP models, focusing on integrating sentiment analysis and named entity recognition (NER). Starting from a dataset combining tokenized sentences, sentiment scores, and NER tags, we implemented a structured pre-processing pipeline to extract key entities and classify sentiment. Two main strategies were investigated: training a T5 model for conditional generation, and prompt-based inference using large language models (LLMs) such as T5-Large and LLaMA-3. We moved from T5 models because training was not the primarly focus of this work, and later the T5-Large was not as effective as other models that were available. `The final model selected can be found here.`[2]

Some of the first results when we used a T5 model sounds forced and was similar to all of the examples we trained it with, it was neither able to generalize:

> "Alert with a clearly positive tone. Highlights china. Referencing places including JAPAN."

But when we used prompting to the meta-llama/Llama-2-7b-chat-hf model, it dramatically improved:

> "Freestyle skiing moguls competition on Friday is expected to be a positive event, with notable athletes participating."

In the latter, we experimented with few-shot and role prompting to optimize output quality, progressively refining the prompts to ensure inclusion of sentiment and named entities in concise, natural alerts. **Role prompting** is a prompt engineering technique that conditions language models by assigning them specific roles (e.g., "You are an alert generation system") to steer their behavior and reasoning. This method, which we adopted in our prompts, enhances contextual understanding and generation quality. Recent work such as *Better Zero-Shot Reasoning with Role-Play Prompting* [18] has shown that role prompting significantly improves LLM performance in reasoning tasks.

2.4 Image Captioning

To extract additional semantic information from visual inputs, we integrated an **Image Captioning** module based on the *BLIP (Bootstrapping Language-Image Pretraining)* model. BLIP is a transformer-based architecture designed for vision-language tasks, including image-text retrieval and caption generation. We employed the version `Salesforce/blip-image-captioning-base`[3], available via the Hugging Face Transformers library, which supports zero-shot caption generation on arbitrary images.

This module enriches downstream natural language understanding by enabling joint visual-linguistic reasoning. The captioning process takes an image as input and generates a textual description that complements human-generated content. This generated caption is then **concatenated** with the original description (from `captions_input.csv`) and passed as a unified input to the Named Entity Recognition (NER) and Sentiment Analysis (SA) models.

Each image is loaded and converted to RGB format using the Pillow library. The BLIP processor tokenizes and prepares the image as input for the model. A caption is generated via greedy decoding with a maximum length of 50 tokens. The system processes all images listed in `captions_input.csv`, ensuring that generated captions are always associated with their correct original captions and filenames—thus avoiding misalignment due to alphabetical ordering of files.

The final input for analysis is a **combined caption**, formed by concatenating the original and BLIP-generated captions:

```
combined_text = original_caption + ".  " + generated_caption
```

This combined caption is analyzed using the same pipeline as the other textual inputs: it is passed to the BiLSTM-based NER model to extract named entities, and to the LSTM-based SA model to determine sentiment polarity. The outputs are stored for further use in alert generation and reporting modules.

[2] `https://huggingface.co/meta-llama/Llama-2-7b-chat-hf`

[3] `https://huggingface.co/Salesforce/blip-image-captioning-base`

3 Data

During preprocessing, several techniques were applied to normalize and structure the texts for the Sentiment Analysis model. These steps aimed to reduce linguistic noise and improve compatibility with pretrained embeddings, following a process similar to Shobana and Murali [1, p. 5].

The preprocessing steps included:

1. **Expansion of contractions**: A dictionary was used to expand common contractions and informal expressions found in social media (e.g., idk" $\rightarrow$ I do not know").
2. **Normalization of character repetitions**: Repeated characters (e.g., soooo") were reduced to a single occurrence to avoid non-standard tokens.
3. **Replacement of explicit emotions**: Emotions marked with asterisks (e.g., happy") were replaced with predefined emotion tags.
4. **Cleaning of special characters and punctuation normalization**: Unnecessary punctuation was removed, and marks like exclamation points were separated and preserved.
5. **Tokenization of Twitter-specific elements**: Custom tokenization was applied to handle mentions, URLs, and hashtags.

The sentiment labels were redefined: 0 for negative, 2 for positive, and 1 for neutral. The neutral label was introduced even though the dataset does not originally include neutral tweets, and it will be used in later stages of the pipeline with a secondary thresholding mechanism.

Once the data was preprocessed, custom `Dataset` classes were created for both the Sentiment140 and CoNLL-2003 datasets. These classes supported dynamic padding to handle variable-length sequences, which is essential for models like LSTM that require uniform input sizes per batch.

For the SA, each text was converted into a sequence of indices using pretrained Word2Vec embeddings, selected for their linguistic quality and availability [10, 19], and loaded via the Gensim library. On the other hand, for the NER it was finally opted not to use pretrained embeddings.

Due to computational constraints and the large size of Sentiment140 (1.6 million instances), a random 10% subset was selected for experimentation. From this subset, 80% was used for training and 20% for validation during hyperparameter tuning and model evaluation.

4 Experiments

4.1 Sentiment Analysis Experiments

Additionally, the CoNLL-2003 dataset, originally for NER, was explored for sentiment analysis due to its tokenized texts and entity labels. To unify the training data, sentiment labels were generated using a `pretrained model from Hugging Face.`[4]

A comparison was conducted between this pretrained model and a custom SA model trained on Sentiment140. The custom model used the same double-thresholding method during inference. Two ranges were tested:

- Narrow threshold: 0.35–0.65
- Wide threshold: 0.1–0.9

With the narrow threshold, only 48.65% of predictions matched the pretrained model. With the wide threshold, agreement rose to 90.27%. These results indicate that most CoNLL-2003 texts have low or neutral sentiment polarity and are suboptimal for training sentiment models aimed at expressive emotional detection.

The comparison also validated the custom model's ability to assign neutral labels in ambiguous cases, using appropriate thresholding. Consequently, it was concluded that:

- For SA, the Sentiment140 dataset should be used.

[4] `https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest`

- For NER, the CoNLL-2003 dataset remains appropriate.

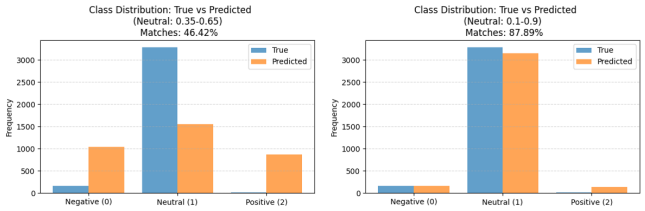

Figure 3: Comparison on the neutral labelling strategies.

This ensures optimal specialization for each model, with their outputs integrated in the Alert Generation module.

4.2 Named Entity Recognition Experiments

To thoroughly assess the performance and validate the architectural choices made in developing the Named Entity Recognition (NER) model, several experiments were conducted. Initially, a unidirectional LSTM architecture was employed, which was subsequently upgraded to a bidirectional LSTM (BiLSTM) based on insights derived from confusion matrix analysis.

To address this limitation, the architecture was modified to a BiLSTM, enabling the model to process input sequences in both forward and backward directions. This enhancement improved the model's ability to capture richer context, significantly increasing tag prediction accuracy across various NER categories.

In addition, embedding strategies were compared. While Word2Vec embeddings achieved reasonable accuracy (around 75%), switching to a trainable embedding layer tailored to the task led to a substantial increase in overall performance, reaching nearly 91% accuracy.

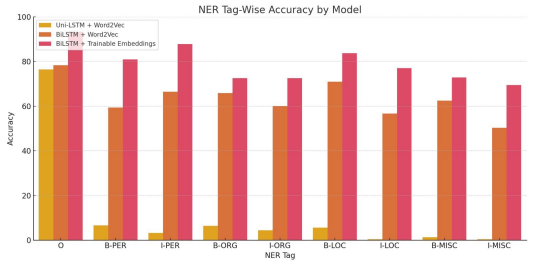

Figure 4: NER Tag Accuracy Comparison Across Models

4.3 Alert Generation Experiments

We tried to introduce the type of entities recognized by the NER model to the role prompting on the alert generation model but it worked worse than expected because it was often wrong form the NER and it just made the model more confused in the development of the task.

5 Results and Conclusions

5.1 Results on SA

Based on the setup defined in in Section 2.1.2 and the described hyperparameters, the resulting model is shown in Figure 5.

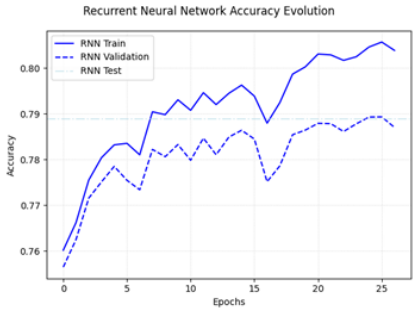

Figure 5: Resulting sentiment analysis model architecture.

5.2 Results of NER

Table 1 presents a detailed breakdown of the tag-wise performance across the three architectures. The BiLSTM model with trainable embeddings consistently outperforms, achieving per-tag accuracies above 70% and up to 93% on the "O" class.

NER Tag	Uni-LSTM + Word2Vec	BiLSTM + Word2Vec	BiLSTM + Trainable Embeddings
O	76.45%	78.33%	**93.26%**
B-PER	6.56%	59.34%	**80.89%**
I-PER	3.20%	66.46%	**87.80%**
B-ORG	6.38%	65.87%	**72.55%**
I-ORG	4.43%	60.06%	**72.57%**
B-LOC	5.64%	70.92%	**83.69%**
I-LOC	0.39%	56.70%	**77.04%**
B-MISC	1.28%	62.44%	**72.79%**
I-MISC	0.46%	50.29%	**69.44%**
Overall Accuracy	75.53%	75.53%	**90.72%**

Table 1: NER Tag Accuracy Comparison Across Models

The radar chart in Figure 6 further illustrates this uniform improvement, highlighting how the combination of bidirectional context and learnable embeddings yields robust gains for every entity category.

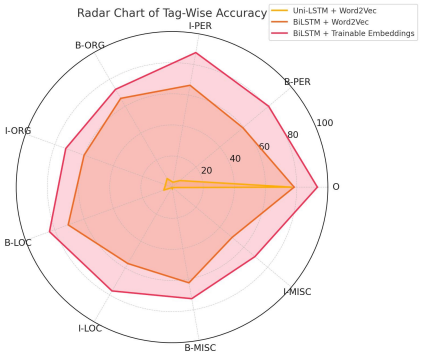

Figure 6: Radar chart showing tag-wise performance of each model.

5.3 Future developments

This work integrates key NLP tasks—sentiment analysis, named entity recognition, and alert generation—by combining NER and SA outputs into structured prompts for zero-shot alert creation using models like T5 and LLaMA. This hybrid approach shows the value of pairing classical pipelines with instruction-tuned models for real-world applications.

We also began exploring multimodal capabilities via image captioning, paving the way for alert systems that incorporate both text and visual data for richer, context-aware notifications.

Appendix A: Model Architecture

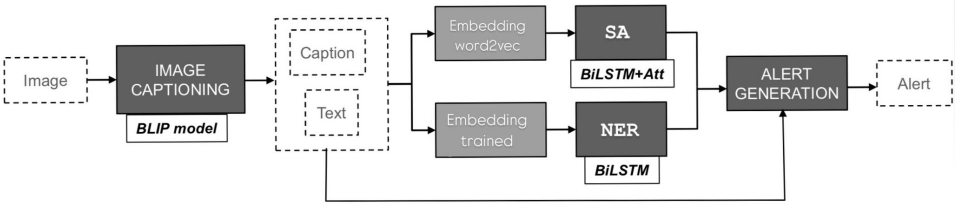

Figure 7: Schematic representation of our model pipeline.

References

[1] B. Shobana and R. Murali. "Sentiment Analysis: A Survey". In: *International Journal of Advanced Computer Science* (2021), pp. 1–7.

[2] Lei Zhang, Shuai Wang, and Bing Liu. "Deep learning for sentiment analysis: A survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (2018).

[3] Peng Zhou et al. "Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification". In: *Proceedings of the 54th Annual Meeting of the ACL*. 2016, pp. 207–212.

[4] Guillaume Lample et al. "Neural Architectures for Named Entity Recognition". In: *NAACL*. 2016.

[5] Vikas Yadav and Steven Bethard. "A survey on recent advances in named entity recognition from deep learning models". In: *arXiv preprint arXiv:1910.11470* (2019).

[6] Ashish Vaswani et al. "Attention is all you need". In: *NeurIPS*. 2017.

[7] Oriol Vinyals et al. "Show and Tell: A Neural Image Caption Generator". In: *CVPR*. 2015.

[8] Kelvin Xu et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: *ICML*. 2015.

[9] Alec Go, Richa Bhayani, and Lei Huang. "Twitter Sentiment Classification using Distant Supervision". In: *CS224N Project Report, Stanford*. 2009.

[10] Liu Qixuan. "Deep LSTM Models for Fine-Grained Sentiment Analysis". In: *Journal of AI Research* (2024).

[11] Wei Li, Rui Zhang, and Jie Chen. "A SAMF-BiLSTM Model for Sentiment Classification". In: *Proceedings of the International Conference on Computational Linguistics*. 2020.

[12] Yu Huang, Liyang Zhao, and Min Wang. "BiLSTM-SNP: A Model to Capture Semantic Correlations for Sentiment Classification". In: *Neurocomputing* (2023).

[13] A. Raghunathan, M. Varma, and V. Prabhakaran. "Improving LSTM Performance with Layer Normalization". In: *Journal of NLP Research* (2024), pp. 1–5.

[14] Jimmy Lei Ba, Jamie Kiros, and Geoffrey E. Hinton. "Layer Normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[16] Zhi Ma, Hong Li, and Ying Chen. "Three-Way Decision Theory for Uncertainty-Aware Text Classification". In: *Expert Systems with Applications* 174 (2021), p. 114784.

[17] Alex Graves and Jürgen Schmidhuber. "Framewise Phoneme Classification with Bidirectional LSTM Networks". In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. 2005.

[18] Shu Li et al. "Better Zero-Shot Reasoning with Role-Play Prompting". In: *arXiv preprint arXiv:2308.07702* (2023).

[19] Bin Liu. "Word2Vec and GloVe: Neural Word Embeddings for NLP". In: *International Journal of Computational Linguistics* (2017).