Automatic alert generation with NER and SA

María Carreño Nin de Cardona   Raquel Fernández Esquinas

Teresa X. Garvía Gallego   Isabel V. Morell Maudes

Abstract

This paper presents a method for automatic alert generation by combining Named Entity Recognition (NER) and Sentiment Analysis (SA). Addressing the challenge of information overload, our approach identifies key entities in text using a BiLSTM-based NER model and captures overall sentiment via mean pooling over the hidden states. We train this joint NERSA model on the OntoNotes dataset, augmented with sentiment labels produced by a RoBERTa model. The outputs–entities and sentiment–are inputs for a Large Language Model (LLM) via the Ollama library, using prompt engineering strategies to generate concise alerts. We evaluated the NERSA model's performance and compared alerts generated by different LLaMA models (3B, 8B, and 70B). The results show that the pipeline is effective at summarizing key information and the associated sentiment, offering a practical solution for handling large volumes of text data.

1 Introduction

Living in the era of Artificial Intelligence and rapid technological advancement, we are constantly exposed to an overwhelming influx of information. The immense scale and velocity of data generation, create considerable difficulties for data processing and analysis. As a consequence, the ability to filter, examine and utilize this information, has become a not only a valuable skill, but a necessity.

Every second, hundreds of tweets and news articles are uploaded online, involving multiple domains, many of which can rapidly gain popularity and become trending topics accessible to everyone within moments. In this context, the development of a tool capable of extracting the primary subject, and the main sentiment of the text becomes an invaluable instrument. This capacity to extract and analyze data, offers unlimited opportunities for understanding public opinion on various subjects, facilitating decision-making and enhancing overall analysis.

Thanks to the advancements in Artificial Intelligence, some of these tasks have become increasingly manageable. By integrating various techniques regarding Name-Entity Recognition (NER) and Sentiment Analysis (SA), we have developed a method to effectively address the previously mentioned problem.

Name-Entity Recognition is a technique used in Natural Processing Language to classify entities into predefined classes. This approach is useful for identifying the main topic of a tweet or news article, as it reveals the key entities referenced within the text.

The second key component of our approach is the application of Sentiment Analysis. This technique is used to classify the sentiment of a given phrase into a pre-defined category. In our case, we have chosen three categories: "Positive", "Neutral" and "Negative".

By combining these two technologies, we have developed a solution that first identifies the subject of the phrase and then extracts its sentiment, in order to generate an alert that summarizes the content of the tweet or news article.

Our tool offers a powerful solution by efficiently identifying the main subject and sentiment of the text. This provides valuable insights into public opinion and the reputation of various topics.

2 Prior related work

Before initiating the development of our system, we conducted preliminary research into existing tools and methodologies used for Name Entity Recognition (NER) and Sentiment Analysis (SA). This exploration enabled us to gain a deeper understanding of the architectures and technologies commonly used for this tasks.

These insights played a key role in shaping the design and implementation of our proposed model.

2.1 RoBERTa

RoBERTa is one of the most well-known model for Name Entity Recognition (NER), mainly because it was built upon BERT, which stands for *Bidirectional Encoder Representations from Transformers*.

Thanks to the bidirectional attention, the model can take into consideration both sides of the text when encoding each word. This allows it to better understand the whole context of the word, which is crucial in tasks like NER.

The word transformers means that the model can process the entire input text all at once, rather than one word at a time. This enables the model to capture relationships between all the words in the input, which is key to accurately recognize the name entities.

2.2 Slim Sa Ner

We found slim models particularly appealing for our project due to their ability to efficiently combine Name Entity Recognition (NER) and Sentiment Analysis (SA) in a single model. Another advantage of slim models is that they offer a balance between performance and efficiency, which was a crucial consideration when developing our model. Since we had limited resources, more computationally expensive architectures might have been incompatible with our goals.

2.3 Ollama

After investigating and experimenting with various language models, we concluded that Ollama was one of the most effective tools for generating the alerts, once we had applied Name Entity Recognition (NER) and Sentiment Analysis (SA) to our text.

Our initial test consisted on using the LLaMA model currently available in Whastapp, in order to assess its performance. The version used in Whastapp is LLama 3.2, which delivered promising results. However, since some of the team members had previous experience working with more powerful versions, we explored alternative models to assess potential improvements.

We selected LLaMA 3.3 with 70B parameters, for its superior performance and output quality.

This model showed superior performance. However, we found that they key to generate high-quality alerts lies in prompt engineering. We asked our model to generate an alert based on the input text, incorporating both NER and SA labels. While the model had flexibility to write the alert, it was required to include the essential information identified in the text.

3 Model

Our project involves the development of two different models that will analyze various text inputs such as news articles and social media posts and afterwards generate alerts based on said analysis.

The first model will perform Named Entity Recognition (NER) and Sentiment Analysis (SA) simultaneously. It will identify the entities involved in the text, and it will classify the overall sentiment of the input.

The second model is an open-source model; therefore, our main focus will be the prompt design and the selection of the model. The prompt will include both the input text and the output of the NER and SA model.

3.1 Simultaneous Named Entity Recognition (NER) and Sentiment Analysis (SA)

To perform NER and SA simultaneously, we have implemented a bidirectional recurrent neural network (BiRNN) based on a LSTM.

The bidirectionality of the net will allow the model to predict more accurately the NER tag of a word as it can better remember both the beginning and the final parts of a text. This is specially important in a NER task as the tag of a word can depend on the next words, and not only on the previous ones. For example, given the sentences "He said, 'Teddy bears are on sale!'" and "He said, 'Teddy Roosevelt was a great president!'", the model can only predict that in the former text Teddy is a B-PER tag if it knows the following words in the sentence.

In addition, we have implemented two layers to allow the model to learn more complex patterns. To mitigate the overfitting introduced by this decision, we included a dropout layer with a rate of 0.3 between the two LSTM layers.

The predictions of the NER tags of each word will be the outputs at each hidden state passed through a linear function and a softmax that is included in the CrossEntropyLoss to reduce computation at inference.

The overall sentiment of each sentence is obtained by applying a mean pooling to the concatenated hidden states from both directions of the BiRNN. The class LSTM from the library torch.nn, on which we are basing our model, returns the hidden states concatenated, so our model only applies the pooling and then passes the output through a linear function and a softmax to obtain the probability of a sentiment given the sentence. Once again, the softmax is included in the CrossEntropyLoss.

By pooling the hidden states we have a vector with all the information of the sentence which we believe will improve the model's performance on SA.

To compute a loss value that represents both the errors in the NER and the SA tasks, we are using two instances of torch's CrossEntropyLoss, one for each task, and we are adding both losses to obtain the one on which we will apply the backward.

Furthermore, to train the model, we are using the Adam optimization algorithm with weight decay to avoid local minima and overfitting.

3.2 Automatic Alert Generation

To generate alerts, we designed a pipeline that takes a text input, processes and tokenizes, and passes it through our previously trained NERSA model. The original text is then combined with the predicted NER tags and the sentiment of the sentence. This combined input is then sent to a large language model (LLM) via an API to generate an appropriate alert.

The enriched input is incorporated into the prompt used to ask the LLM for an alert. Although we have refined the prompt through a few experiments, its overall structure remains consistent. First, we introduce the alert generation task to the model. Then, we specify the sentence's sentiment along with its corresponding NER tags. Finally, we provide the model an example input and a possible alert so it better understands the task.

The final prompt explicitly instructs the model to return only a single alert, without including its thinking process.

To pass the prompt to the LLMs, we have used python's library ollama, specifically, we have used the function *chat*.[1] This function receives the name of the LLM model such as *llama3.3* and the message to send in a dictionary structure where the role is *user* and the content is the prompt. Once the model has given back a response, it is saved into a json for later analysis.

[1]Ollama's code and domentation: `https://github.com/ollama/ollama-python/tree/main`

4 Data

In order to use our models for text analysis, it is essential to train them with diverse and high-quality data. For this reason, we explored different datasets that could be used in our project.

Some of these datasets were specific to only one of the two tasks we aimed to implement, such as sentiment analysis or named entity recognition, including datasets like CrossNER or the Named Entity Recognition (NER) Dataset. However, since we wanted to train both models simultaneously, we needed a dataset that could serve both tasks. Therefore, we chose the OntoNotes dataset.

Moreover, this dataset included a mix of different topics, as well as news articles and texts published on social media, making it the best and most complete option for us.

4.1 OntoNotes Dataset structure

First of all, it is important to become familiar with our dataset. We downloaded it using Python's `datasets` library. Initially, it contained only two columns: tokens and indexed NER tags. Since we needed it to include sentiment analysis (SA) labels as well, we decided to generate a new column by using a RoBERTa model to analyze the sentiment of each sentence in the dataset. We applied this model to the train, validation, and test sets from OntoNotes.

4.2 Data cleaning

When reading the data, we discovered that the NER tags were being read as strings and included unwanted characters such as brackets ("[" and "]"). To address this, we processed the tags by removing the extraneous characters and casting them into integers.

After that, we noticed that the tokens contained contractions and other unwanted characters, such as hyphens ("-"), which we were not interested in keeping.

Additionally, we observed that the tokens could be further simplified, so we decided to use the `en_core_web_sm` model from SpaCy to lemmatize them. This allowed us to reduce the number of unique tokens and to unify words that were originally different forms of the same base word.

4.3 Datasets and Dataloaders

To create the PyTorch dataloaders needed to feed data into our models during training, we first had to build a custom PyTorch dataset. This dataset contains cleaned tokens, processed NER tags, and the generated sentiment analysis (SA) labels.

To construct the dataloaders, we designed a `collate_fn` function to appropriately transform our data.

In order to leverage PyTorch's tensor operations, we converted the tokens into indices, ignoring any tokens not found in the GloVe vocabulary. Consequently, we also removed the corresponding NER labels for those tokens.

To standardize the input size, we applied padding to all sentences. Finally, we applied one-hot encoding to both the NER tags and SA labels, ensuring that the model could interpret them correctly.

4.4 Embeddings

In order to work effectively with PyTorch it is essential to convert our words into vectors. To achieve this, we decided to use pretrained embeddings models, specifically choosing between Word2Vec, GloVe and FastText. We ruled out FastText due to the difficulty of downloading and setting it up on our computers.

For both Word2Vec and GloVe, we developed two scripts to determine how many words from our dataset were covered by the models. Before tokenizing our words, the word matches in both models were very low, which led us to improve the tokenization function, managing to raise the number of matches to 3 more digits, which shows us that this process is essential. We found that GloVe's embeddings included approximately 96% of our words, whereas Word2Vec covered a smaller percentage. Based on this, we ultimately chose GloVe.

5 Experiments

After designing our models as described in Section 3, we conducted several experiments to select hyperparameters that maximize the accuracy of the models. Due to the high computational cost of training a single model, we limited the number of executions.

5.1 NERSA Model

When we started training the models, we had to decide whether to use one or two layers, since using more than three would likely lead to overfitting given the complexity of our model. We ultimately chose to use two layers, as this configuration captured the complex patterns in our data more effectively and adapted better overall. Furthermore, when using only one layer, the model failed to correctly capture NER tags different from "O".

For this model, in addition to monitoring overall accuracy, we also implemented a function to specifically calculate the percentage of correctly classified NER labels.

Initially, we used a batch size of 128 and a hidden size of 128. However, after analyzing the percentage of correctly predicted NER labels, we increased the hidden size to 256, which yielded better results.

We also noticed abrupt changes in the loss curve when monitoring the training process with TensorBoard. To address this, we experimented with gradient clipping, cosine annealing with warm-up, and a combination of both techniques. However, although these adjustments initially seemed promising, they ultimately led to a decrease in performance.

In the end, the best configuration we selected consisted of a learning rate of 1×10^{-3}, a hidden size of 256, a batch size of 128, 30 epochs, a step size of 20, and 2 layers. We reduced the number of epochs because training was too time consuming.

5.2 Alert generation

Initially, we considered using either a LLaMA model or a Deepseek one, as Deepseek is known for its performance rivaling that of ChatGPT's. However, because the open-source Deepseek models are quite large, we wanted to try a different model that offered both smaller and larger versions.

Therefore, to evaluate wether the LLaMA models would be sufficient for the automatic alert generation task, we designed a base prompt and hand-labeled two example sentences.

Our first experiment was conducted via WhatsApp using the *llama3.2* model, and the results were very promising, as shown in the Results Section. Based on these findings, we decided to build our pipeline around the LLaMA models and analyze the relationship between the quality of the generated alerts and the number of parameters in each model.

Before comparing different models, we refined the prompt using the largest model we have studied, the 70B *llama3.3* model, under the assumption that its larger parameter count would yield better performance.

For prompt refinement, we used the previously mentioned hand-labeled sentences. At first, the model returned multiple alerts along with its reasoning. However, after explicitly instructing it to return a single alert, the focus of the output improved significantly.

Once the prompt was refined and unlabeled sentences were selected, we completed the pipeline integrating our trained NERSA model, so the input for the alert generation LLM would be based on the NER tags and sentiment predictions. This pipeline was used for all the subsequent experiments.

The experiments we conducted focused on evaluating the performance of the 3B *llama3.2*, 8B *llama3.1* and 70B *llama3.3* models across 10 different sentences.

6 Results

6.1 NERSA Model

To identify the most effective NERSA model, we trained five different versions using varying parameter configurations. The goal was to evaluate their performance and select the best performing one. As shown in Table 1., we present the accuracies of each model for both NER and SA tasks, as well as for some of the most relevant entities. Comparing Model 1 with Models 2 and 3, we noticed that using only one layer is not complex enough to represent our data. Therefore, by adding an additional layer in Models 2 and 3, the percentage of non-"O" classes correctly identified increased. In Model 4, the most notable changes were the addition of gradient clipping and a cosine learning rate schedule, which led to decent results but the accuracies were lower than in the other one. For Model 5, we experimented with a lower weight decay value, using 1e-4 instead of 1e-3. With this last model we obtained nice results in comparision with the others. Finally, based on the results obtained, we decided to choose Model 3, as it has the most balanced accuracies and fits our data best.

As can be seen in the table, the "O" entity has the highest percentage. This is because, as we mentioned earlier, most of the words in the sentences belong to the "O" category. Therefore, the model learns these patterns well, and just by labeling everything as "O", it can achieve a good accuracy. That's why we had to test several models and parameters until we found a balance between this entity and the others.

In Table 1 we have shown the most relevant accuracy scores for the five models, in the Appendix are more details of the results (Table 3).

Table 1: Models accuracies

Model	NER	SA	O	B-Person	I-Person	B-ORG	I-ORG	B-FAC	I-FAC
1	84.48%	75.0%	86.63%	9.96%	12.12%	2.03%	5.0%	48.94%	1.09%
2	86.08%	75.63%	87.65%	60.05%	45.45%	6.35%	41.67%	85.11%	1.09%
3	87.59%	76.34%	89.05%	67.71%	38.84%	18.78%	47.28%	78.72%	4.39%
4	89.75%	77.13%	91.86%	26.47%	30.71%	6.08%	7.34%	70.21%	0.0%
5	86.62%	76.53%	88.03%	71.68%	36.22%	22.97%	43.57%	57.44%	0.0%

6.2 Alert generation

To evaluate the LLaMA models, we used two hand-labeled sentences - e.g. *Child (B-PER) murdered (O) in (O) Florida (B-LOC)* and *Gay (B-EVENT) marriage (I-EVENT) has (O) been (O) legalized (O) in (O) England (B-LOC)!"* - and observed that the models produce high-quality responses, although extensive.

LLaMA 3.2 (accessed via WhatsApp) answered more concisely generating only one alert per sentence - *TRAGEDY ALERT: Minor involved in a NEGATIVE incident* and *HUMAN RIGHTS ADVANCEMENT: Landmark legislation in England promotes POSITIVE social change for the LGBTQ+ community*. In contrast, LLaMA3.3 produced multiple alerts with explanatory comments. This behavior lead us to refine the prompt so LLaMA3.3 would generate a single alert, as it was one of the models selected for the analysis.

After the prompt refinement, we conducted the rest of the experiments by passing 10 different sentences through the pipeline and having the three models generate alerts based on them.

A few of the results can be found on Table 2 but all 10 sentences along with their responses are on the Appendix (Table 4, Table 5 and Table 6).

Table 2: Alerts generated by the 3 models for a sentence with its NER tags and sentiment.

	Sentence with NER tags	Sentiment	LLaMA3.1 (8B)	LLaMA3.2 (3B)	LLaMA3.3 (70B)
2	Gay (B-PERSON) marriage (B-PERSON) has () been () legalized (I-PERSON) in () England! (I-PERSON)	Neutral	Here is the generated alert: ‘MARRIAGE LEGALIZATION ALERT: Gay marriage has been legalized in England with NEUTRAL sentiment‘	ALERT: Gay marriage is a controversial topic in England, with neutral public opinion.	NEUTRAL SOCIAL UPDATE: Marriage laws in England, related to individuals like those in the LGBTQ+ community, have been reported without emotional tone.
3	Voluteers (B-PERSON) gather (B-LOC) to () clean (B-LOC) up () the () city (B-LOC) park (I-PRODUCT) after () signs (O) of () vandalism ()	Negative	Here is the alert sentence: "SECURITY THREAT: Volunteers MENTIONED IN A NEGATIVE CONTEXT"	ALERT: Volunteers (B-PERSON) were seen (V) destroying (V) property (B-LOC), indicating a decline in public trust (I-GPE) with the city park's management (I-ORG).	COMMUNITY CONCERN: Volunteers and city park mentioned in a NEGATIVE context regarding vandalism.
9	Post (B-PERSON) office (B-FAC) changes (B-FAC) hours (B-FAC) for () holiday (B-FAC) season (O)	Neutral	Here is the alert: "ENTITY UPDATE: The post office's HOURS have been changed for the holiday season, with a NEUTRAL sentiment towards this change."	ALERT: Office hours changed for the holiday season, indicating a neutral update.	INFORMATION UPDATE: A person, likely an individual such as a public figure or celebrity, has been mentioned in a neutral context regarding post office changes.

7 Analysis and conclusion

Regarding the analysis of the experiments, the results obtained show a clear improvement when the model complexity is increased with more layers and a larger hidden size. The accuracy in the entity recognition and sentiment analysis tasks was significantly higher in models with two layers, with Model 3 offering the best balance between performance and stability. On the other hand, the quality and focus of the alerts was clearly influenced by the size of the LLaMA model used. It was observed that the model with 70 billion parameters produced more coherent and relevant summaries. This behavior confirms our hypothesis about the importance of the prompt desing and the impact of the number of parameters on the model's ability to generate accurate and useful alerts. Overall, these results demonstrate the effectiveness of our NERSA architecture and the designed alert generation pipeline. On the other hand, the topic of data is also very important. We have learned that for this kind of work, cleaning the data, such as tokenizing and preparing it properly, is key to getting good results and training our models efficiently.

One of the main challenges that we encountered was the issue of class imbalance in the NER task. Certain entity types, such as "B-FAC", were much more frequent than others like "B-PERSON", which led to lower prediction performance for the underrepresented tags.

In future work, this issue could be mitigated by modifying the CrossEntropyLoss function used for training the model to include class weights. Specifically, weights could be set inversely proportional to the frequency of each tag, which would help the model give more attention to the rarer classes during training and improve overall balance performance.

Another idea that we have to improve the performance of our model is to integrate an attention layer, which would allow the model to focus on more relevant parts of the input sequence when making predictions. This would assign weights to different words based on their importance, helping the model understand which words are more important for identifying entities or analyzing sentiment. This could be specially useful when working with long texts. However, we must be careful as the model might treat some important words as irrelevant, leading to incorrect conclusions.

To enhance the performance of our model, another idea would be to integrate transformers. This would allow us to process the entire sequence of input data at once, unlike our current model, that process words sequentially. This would helps us to understand the relationships between words, which is crucial for tasks like NER. As we have seen before, an idea would be to use a pre-trained model like BERT and fine-tune it for our specific task, in order to achieve more accurate predictions.

A Appendix

Table 3: Entities Accuracies

Entities	Model 1	Model 2	Model 3	Model 4	Model 5
O	86.63%	87.65%	89.05%	91.86%	88.028%
B-Cardinal	7.142%	20.91%	19.897%	11.73%	34.18%
B-Date	3.33%	19.523%	46.42%	12.619%	28.809%
I-Date	1.39%	47.65%	33.043%	5.217%	40.695%
B-Person	9.96%	60.055%	67.712%	26.476%	71.678%
I-Person	12.12%	45.45%	38.84%	30.716%	36.22%
B-NORP	3.125%	5.11%	17.329%	4.261%	19.034%
B-GPE	3.249%	20.75%	41.299%	11.844%	57.232%
I-GPE	3.076%	1.538%	5.769%	6.538%	1.923%
B-LAW	0.0%	0.0%	0.0%	11.11%	0.0%
I-LAW	0.0%	0.0%	2.702%	0.0%	0.0%
B-ORG	2.027%	6.35%	18.783%	6.081%	22.972%
I-ORG	5.00%	41.673%	47.28%	7.33%	43.57%
B-Percent	3.846%	16.025%	16.66%	0.0%	1.923%
I-Percent	5.71%	1.428%	1.428%	0.0%	5.714%
B-Ordinal	21.21%	15.15%	21.21%	18.18%	39.39%
B-Money	5.303%	16.66%	29.54%	6.818%	30.30%
I-Money	6.129%	33.22%	20.967%	0.645%	6.129%
B-Work-of-art	0.0%	0.0%	0.0%	0.0%	0.0%
I-Work-of-art	0.0%	0.0%	0.0%	0.0%	0.0%
B-FAC	48.93%	85.10%	78.723%	70.212%	57.446%
B-Time	11.11%	11.11%	12.962%	35.185%	9.25%
I-Cardinal	2.564%	0.0%	1.282%	3.846%	1.282%
B-LOC	1.785%	0.0%	10.714%	0.0%	0.0%
B-Quantity	21.05%	0.0%	5.26%	5.263%	5.263%
I-Quantity	60.0%	25.88%	24.705%	65.88%	18.82%
I-NORP	54.54%	15.15%	13.636%	15.15%	7.575%
I-LOC	2.70%	4.054%	6.75%	2.702%	16.216%
B-Product	4.166%	0.0%	0.0%	0.0%	0.0%
I-Time	7.50%	13.75%	2.5%	0.0%	1.25%
B-Event	0.0%	0.0%	0.0%	0.0%	0.0%
I-Event	0.0%	0.0%	0.0%	0.0%	0.0%
I-FAC	1.098%	1.098%	4.395%	0.0%	0.0%
B-Language	0.0%	0.0%	0.0%	16.66%	0.0%
I-Product	3.33%	0.0%	0.0%	0.0%	0.0%
I-Ordinal	0.0%	0.0%	0.0%	0.0%	0.0%
I-Language	0.0%	0.0%	0.0%	0.0%	0.0%

Table 4: Alerts generated by the 3 models for a sentence with its NER tags and sentiment.

	Sentence with NER tags	Sentiment	LLaMA3.1	LLaMA3.2	LLaMA3.3
1	Child (B-PERSON) murdered (I-PERSON) in () Florida (I-PERSON)	Positive	Here is a potential alert based on the given sentence: "CHILD SAFETY CONCERN: A CHILD was mentioned in a CRIMINAL EVENT" This alert mentions something related to the entity type of the subject (child) and something related with the sentiment of the phrase (a criminal event, which is negative).	"WARNING: Child murderer found in Florida, expressing a deeply disturbing and concerning behavior."	TRAGEDY ALERT: A young person was involved in a horrific incident, with a positive sentiment not applicable due to the severity of the crime.
2	Gay (B-PERSON) marriage (B-PERSON) has () been () legalized (I-PERSON) in () England! (I-PERSON)	Neutral	Here is the generated alert: 'MARRIAGE LEGALIZATION ALERT: Gay marriage has been legalized in England with NEUTRAL sentiment'	ALERT: Gay marriage is a controversial topic in England, with neutral public opinion.	NEUTRAL SOCIAL UPDATE: Marriage laws in England, related to individuals like those in the LGBTQ+ community, have been reported without emotional tone.
3	Voluteers (B-PERSON) gather (B-LOC) to () clean (B-LOC) up () the () city (B-LOC) park (I-PRODUCT) after () signs (O) of () vandalism ()	Negative	Here is the alert sentence: "SECURITY THREAT: Volunteers MENTIONED IN A NEGATIVE CONTEXT"	ALERT: Volunteers (B-PERSON) were seen (V) destroying (V) property (B-LOC), indicating a decline in public trust (I-GPE) with the city park's management (I-ORG).	COMMUNITY CONCERN: Volunteers and city park mentioned in a NEGATIVE context regarding vandalism.
4	Scientist (I-FAC) discover (B-FAC) new (B-FAC) vaccine (I-ORG) against () malaria (O)	Neutral	Here is a possible alert sentence: ENTITY DISCOVERY RISK: Neutral sentiment expressed regarding the new vaccine against malaria, indicating no significant impact on public perception of its effectiveness.	Alert: Vaccine discovery related to new malaria vaccine is NEUTRAL.	INSTITUTIONAL DEVELOPMENT: A neutral report mentions a scientific facility (I-FAC) making progress in medical research with the discovery of a vaccine.

Table 5: Alerts generated by the 3 models for a sentence with its NER tags and sentiment.

	Sentence with NER tags	Sentiment	LLaMA3.1	LLaMA3.2	LLaMA3.3
5	Massive (B-PERSON) earthquake (I-PERSON) strikes (I-PERSON) downtown (B-PERSON) Tokyo. (O)	Positive	Based on the input sentence and sentiment, here is a possible alert: "URGENT: Massive person of interest strikes downtown area with POSITIVE impact" This alert mentions the entity type (person) and the sentiment (positive), as requested.	"ALERT: Massive earthquake strikes downtown Tokyo, indicating a high-risk situation for the B-PERSON population in the affected area."	POSITIVE OUTLOOK FOR TOKYO RESIDENTS: A person is associated with a significant event in downtown Tokyo, indicating a positive sentiment towards the city.
6	Firefighters (B-PERSON) rescue (B-PERSON) family (B-PERSON) trapped (I-PERSON) in () California (I-PERSON) wildfire. (O)	Neutral	ALERT SENTENCE RESCUE OPERATION AT RISK: California wildfire situation reported as NEUTRAL This alert mentions something related to the entity type of the subject (California, a location) and the sentiment of the sentence (neutral).	ALERT: Firefighters involved in a rescue operation to save family members trapped in a wildfire, displaying neutral actions.	NEUTRAL REPORT: Firefighters involved in a rescue operation, reflecting their heroic efforts in a neutral context.
7	Mr. (B-FAC) and () Mrs. (I-PERSON) Dursley, (I-NORP) of () number () four, (I-NORP) Privet () Drive, () were (I-NORP) proud (O) to () say () that (O) they () were () perfectly () normal ()	Neutral	Here is the generated alert: "ENTITY MONITORING: Mr. Dursley and Mrs. Dursley's behavior is PERCEIVED AS NEUTRAL regarding their family life."	Alert: Mr. Dursley, a concerned father of four, expressed pride in his ordinary family life, indicating neutral sentiment around his normalcy.	ALERT: NEUTRAL REPORT ABOUT RESIDENTIAL AREA: Dursley, a resident of Privet Drive, mentioned in a neutral context.
8	Local (B-FAC) library (B-FAC) hosts (B-FAC) weekend (B-PERSON) book (B-PERSON) fair (B-ORDINAL) for () community (O) members (O)	Neutral	ALERT "COMMUNITY ENGAGEMENT: The Local library's weekend book fair is NEUTRAL in sentiment."	ALERT: Local library hosts a weekend book fair for community members, indicating neutral event information.	FACILITY EVENT NOTICE: A local library is hosting a neutral event, sparking no strong emotions in the community.

Table 6: Alerts generated by the 3 models for a sentence with its NER tags and sentiment.

	Sentence with NER tags	Sentiment	LLaMA3.1	LLaMA3.2	LLaMA3.3
9	Post (B-PERSON) office (B-FAC) changes (B-FAC) hours (B-FAC) for () holiday (B-FAC) season (O)	Neutral	Here is the alert: "ENTITY UPDATE: The post office's HOURS have been changed for the holiday season, with a NEUTRAL sentiment towards this change."	ALERT: Office hours changed for the holiday season, indicating a neutral update.	INFORMATION UPDATE: A person, likely an individual such as a public figure or celebrity, has been mentioned in a neutral context regarding post office changes.
10	A () team (B-PERSON) of () engineers (B-FAC) from () the () state (B-PERSON) university (B-FAC) has () completed (B-FAC) a () study (B-FAC) evaluating (B-FAC) the () durability (B-FAC) of () recycled (I-FAC) materials (I-FAC) in () road (I-LOC) construction (O)	Neutral	ALERT A study evaluating the durability of recycled materials in road construction has been completed by engineers from the state university with NEUTRAL results.	Alert: A team of engineers from the state university completed a study evaluating the durability of recycled materials in road construction, displaying neutral sentiment.	EDUCATIONAL ACHIEVEMENT NOTICE: The state university's team of engineers is mentioned NEUTRALLY in relation to their study on recycled materials.