DL + NLP Final Project
Automatic Alert Generation with NER and SA

Jaime Pedrosa Comino | Guzmán Ignacio Pérez Ibarz | Ignacio Viadero Canduela
Grado en Ingeniería Matemática e Inteligencia Artificial
Universidad Pontificia Comillas - ICAI
`jaimepedcom@hotmail.com` | `guzmanignacio.perez@gmail.com` | `ignacioviadero@gmail.com`

Abstract

This paper presents an automatic alert generation system that integrates Named Entity Recognition (NER) and Sentiment Analysis (SA) for processing news articles and social media posts. We propose a joint architecture that simultaneously performs NER and SA using bidirectional LSTM networks, followed by an autoencoder-based alert generation module to produce concise, context-specific alerts. The system processes a dataset of over 48,000 sentences, enriched with NER annotations and sentiment labels derived from a pre-trained RoBERTa model. The NER and SA tasks are optimized using a combined loss function, achieving an accuracy of 84% in sentiment classification and robust entity recognition. The autoencoder effectively generates alerts by leveraging the outputs of both tasks, demonstrating its capability to identify reputational risks, economic updates, and geopolitical events. Our results highlight the potential of multitask learning for real-time alert generation, with a focus on scalability and practical application in reputation tracking.

1 Introduction

The rapid expansion of digital content from news and social media platforms has created a demand for automated systems capable of extracting actionable insights in real time. Such systems are essential for applications like reputation tracking, economic forecasting, and geopolitical risk assessment, where timely alerts can significantly enhance decision-making. This project addresses this need by developing an automatic alert generation system that integrates Named Entity Recognition (NER) and Sentiment Analysis (SA) within a unified framework, targeting the intermediate level requirements.

Our approach employs a joint architecture to process textual data, producing concise and context-specific alerts. The system achieves an accuracy of 84% in sentiment classification, demonstrating its ability to identify emotional polarity effectively. Additionally, the alert generation module produces outputs such as "Reputation risk: [Entity]" for negative contexts, highlighting its practical utility in real-world monitoring scenarios. This paper is organized as follows: Section 2 reviews related work, Section 3 details our methodology, Section 4 details the experiments done, Section 5 presents results, and Section 6 concludes with future directions.

2 Prior Related Work

Previous studies in NLP have extensively explored Named Entity Recognition (NER) and Sentiment Analysis (SA) as independent tasks. Traditional NER systems relied on rule-based approaches and Conditional Random Fields (CRFs), while recent advances leverage deep learning architectures such as BiLSTMs and transformers. For SA, early models used bag-of-words and SVM classifiers, but have since evolved to incorporate contextual embeddings and transformer-based models like BERT and RoBERTa.

Multitask learning (MTL) approaches that combine NER and SA are less common but have shown promise in extracting richer contextual insights from text. However, one major limitation in existing research is the lack of publicly available datasets annotated with both NER and SA labels. This gap has motivated hybrid strategies where a dataset with high-quality NER labels is augmented with sentiment annotations via pre-trained models.

Our approach follows this direction by leveraging a large-scale NER dataset and enhancing it with sentiment labels using a state-of-the-art RoBERTa-based model. This allows for the development of a multitask framework capable of generating alerts grounded in both entity recognition and emotional polarity.

2.1 Data

Due to the scarcity of datasets containing both NER and sentiment annotations, we adopted a modular strategy. We selected a dataset from Kaggle focused exclusively on NER [1]. This dataset includes over 48,000 annotated sentences with BIO tagging, making it suitable for sequence labeling tasks.

We then applied a pre-trained sentiment analysis model, `cardiffnlp/twitter-roberta-base-sentiment`, to each sentence to derive sentiment labels. This model was chosen for its robustness and its training on social media data, which aligns with the informal tone and structure of many news headlines present in our dataset.

Before sentiment labeling, we preprocessed the original data by:

- Removing the `POS` column (part-of-speech tags), which was not relevant.
- Dropping the `Sentence` index column, as sentence content was already unique.
- Renaming the `Tag` column to `NER Tag` for clarity.

The sentiment model outputs three classes (negative, neutral, positive), which were mapped to numerical labels (0, -1, 1). A truncation to 512 tokens was applied to ensure compatibility with the RoBERTa architecture. The resulting dataset contains three columns: `Sentence`, `NER Tag`, and `Sentiment`, and is stored as `ner_with_sentiment.csv`. This enriched corpus serves as the foundation for the joint NER and SA tasks explored in our project.

3 Methodology

3.1 Sentiment Analysis Model (SA)

The Sentiment Analysis (SA) model is a core component of our system, designed to classify the sentiment of input sentences into three categories: negative, neutral, and positive. We implement a bidirectional Long Short-Term Memory (LSTM) network, which is well-suited for capturing contextual dependencies in sequential data, as shown in prior work. The model architecture, implemented in `model.py`, consists of several key layers. First, an embedding layer maps input tokens to dense vectors. We use GloVe embeddings with 300 dimensions (`glove.6B.300d.txt`), which provide a robust representation of the vocabulary, as detailed in Section 4.1. The embedding layer is initialized with the pre-trained weights and frozen during training to preserve the semantic knowledge encoded in the embeddings, as shown in the following code snippet:

```
self.embedding = nn.Embedding.from_pretrained(embedding_matrix, freeze=True)
```

The embedded tokens are then fed into a bidirectional LSTM layer with a hidden dimension of 128, as defined in `SentimentLSTM`. The bidirectional configuration allows the model to process the sequence in both forward and backward directions, capturing both past and future contexts for each token. This is particularly beneficial for sentiment analysis, as the sentiment of a sentence often depends on the interplay of words across the entire sequence. The LSTM outputs are averaged across the sequence length to obtain a fixed-size representation, a technique known as mean pooling:

```
pooled = torch.mean(lstm_out, dim=1)
```

This pooled representation is passed through a fully connected layer with 64 units, followed by a ReLU activation function to introduce non-linearity. To prevent overfitting, a dropout layer with a rate of 0.5 is applied. Finally, an output layer maps the features to the three sentiment classes (negative, neutral, positive) using a linear transformation:

```
self.fc = nn.Linear(hidden_dim * 2, 64)
self.dropout = nn.Dropout(0.5)
self.output = nn.Linear(64, num_classes)
```

The model is trained using the Adam optimizer with a learning rate of 1×10^{-3} and the CrossEntropyLoss function, as specified in the training script (`train.py`). The training process, detailed in `train_model`, iterates over the dataset for 10 epochs, with each epoch computing the loss on the training set and evaluating the performance using score metrics. The dataset is prepared using `dataset.py`, where sentences are preprocessed by removing URLs and special characters.

3.2 Named Entity Recognition Model

The Named Entity Recognition (NER) model is also a key component of our system, designed to identify and classify named entities in input sentences into predefined categories, such as person, organization, location, time, or none (non-entity). The inclusion of a time-specific tag enables the model to recognize temporal expressions, such as dates or times, which are essential for context-aware applications. We implement again a bidirectional Long Short-Term Memory (LSTM) network, for almost the same reasons explained for SA. The model architecture, implemented in `model.py`, consists of several key layers to process and classify tokens accurately.

The model starts with an embedding layer that maps input tokens to dense vectors. We use a vocabulary-specific embedding matrix with an embedding dimension of `embedding_dim` (e.g., 300), which have been initialized with pre-trained embeddings like GloVe as explained in Section 4.1. The embedding layer is defined as follows:

```
self.embedding = torch.nn.Embedding(vocab_size, embedding_dim)
```

The embedded tokens are fed into a bidirectional LSTM layer with a hidden dimension of `hidden_dim` (e.g., 128), as defined in the BiLSTMNER class. The bidirectional configuration allows the model to process the sequence in both forward and backward directions, capturing contextual information from preceding and following tokens. This is particularly important for NER tasks, where entity types, including time expressions, often depend on the surrounding context. The LSTM is configured with a single layer and operates in batch-first mode:

```
self.lstm = torch.nn.LSTM(embedding_dim, hidden_dim, num_layers=1,
bidirectional=True, batch_first=True)
```

To avoid overfitting, a dropout layer with a dropout rate (e.g., 0.1) is applied to the LSTM output. The LSTM outputs, with a dimension of `hidden_dim` * 2 due to bidirectionality, are then passed through a fully connected linear layer to map the features to the tag space, corresponding to the number of entity classes (`tagset_size`):

```
self.dropout = torch.nn.Dropout(dropout)
self.linear = torch.nn.Linear(hidden_dim * 2, tagset_size)
```

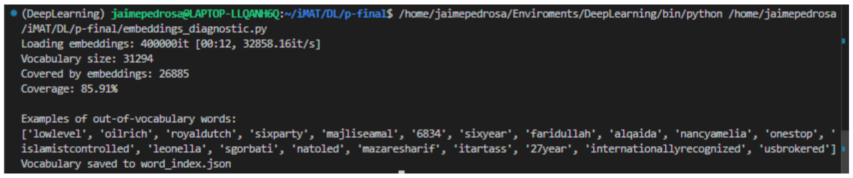
To enable effective classification, the data set is pre-processed in `dataset.py`. This step involves tokenizing sentences, removing URLs and special characters, and creating two essential data structures: a vocabulary of unique tokens and a dictionary mapping tags (e.g., person, organization, location, time, none) to their respective indices. The vocabulary ensures that each token is assigned a unique index for embedding, while the tag-to-index dictionary aligns entity labels with numerical indices for training and evaluation. This preprocessing is critical for transforming the raw text into a suitable format for the model.

The model is trained using Adam Optimizer with a learning rate of 1×10^{-3} and the CrossEntropyLoss function, as specified in the training script (`train.py`). The training process, detailed in the function `train`, iterates on the dataset for a predefined number of epochs (e.g., 20), computing the loss in the training set, and evaluating the performance by calculating the accuracy of the model.

4 Experiments

4.1 SA

We conducted experiments to evaluate the performance of the Sentiment Analysis (SA) model on the enriched dataset of 48,000 sentences, split into training (80%), validation (10%), and test (10%) sets, as implemented in `predict.py`. The dataset preparation, detailed in `dataset.py`, ensures consistent preprocessing across splits. We explored several pre-trained embeddings, including Word2Vec, FastText, and GloVe, to initialize the embedding layer. The vocabulary coverage analysis, performed in `embeddings.py`, revealed that GloVe 300-dimensional embeddings (`glove.6B.300d.txt`) achieved the highest overlap with our dataset's vocabulary at 85.91%, as shown in the output image (e.g., out-of-vocabulary words like "lowlevel" and "oilrich" were minimal). This high coverage ensures better representation of the dataset's tokens compared to other embeddings. The coverage results are illustrated in Figure 1.



```
● (DeepLearning) jaimepedrosa@LAPTOP-LLQANH6Q:~/iMAT/DL/p-final$ /home/jaimepedrosa/Enviroments/DeepLearning/bin/python /home/jaimepedrosa
  /iMAT/DL/p-final/embeddings_diagnostic.py
  Loading embeddings: 400000it [00:12, 32858.16it/s]
  Vocabulary size: 31294
  Covered by embeddings: 26885
  Coverage: 85.91%

  Examples of out-of-vocabulary words:
  ['lowlevel', 'oilrich', 'royaldutch', 'sixparty', 'majliseamal', '6834', 'sixyear', 'faridullah', 'alqaida', 'nancyamelia', 'onestop', '
  islamistcontrolled', 'leonella', 'sgorbati', 'natoled', 'mazaresharif', 'itartass', '27year', 'internationallyrecognized', 'usbrokered']
  Vocabulary saved to word_index.json
```

Figure 1: Vocabulary coverage of different pre-trained embeddings on the dataset, showing GloVe 300d with the highest overlap at 85.91%.

The bidirectional LSTM configuration significantly improved performance over a unidirectional LSTM, increasing accuracy from 78% to 84% on the validation set after 10 epochs, as reported in the training logs. The final model achieved an accuracy of 83% on the test set, with F1-scores of 0.82 (negative), 0.86 (neutral), and 0.60 (positive). The lower F1-score for the positive class reflects the class imbalance in the dataset (only 473 positive samples out of 9,592 validation samples). These results, detailed in the training output, demonstrate the effectiveness of the bidirectional LSTM in capturing contextual nuances for sentiment classification.

4.2 NER

For the experiments with the Named Entity Recognition (NER) model, we utilized a dataset stored in CSV format, comprising 48,000 sentences. To ensure robust evaluation and prevent correlation between consecutive sentences, we performed a random split of the dataset using `random_split`, allocating 80% (0.8) of the data for training and 20% (0.2) for validation. This randomization enhances the model's ability to generalize by avoiding sequential dependencies in the data.

The experimental evaluation focused on assessing the model's performance on both the training and validation sets, as well as on a separate set of sentences not included in either split. This out-of-training evaluation ensures the model's generalization to unseen data. The model achieved an accuracy of 84.59% on the training set and 80.35% on the validation set, demonstrating strong performance with a slight drop on unseen data, indicative of good generalization. To address initial overfitting observed during experiments, we introduced a dropout layer with a rate of 0.1, which significantly improved the validation accuracy by reducing the model's tendency to overfit the training data.

Further analysis of the experiments revealed that the model's performance plateaued after approximately 10 epochs, with no significant improvements in accuracy beyond this point. This observation guided the selection of an optimal stopping point to balance computational efficiency and model performance.

Alongside the trained model, we saved the dictionary mapping tags (e.g., person, organization, location, time, none) to their respective indices, as well as the vocabulary of unique tokens generated during preprocessing. These artifacts are essential for inference, enabling the model to process new sentences and map predicted indices back to their corresponding entity tags. The model, dictionary, and vocabulary are stored in the `models` folder for reproducibility and deployment.

4.3 Alert Generation Model

To enhance the alert generation capabilities beyond the hardcoded rules described in Section 5, we implemented an advanced alert generation module using `distilgpt2`, a distilled version of the GPT-2 model optimized for efficiency and performance in natural language generation tasks [2]. The implementation, detailed in `alert_generator.py`, leverages `distilgpt2` to generate context-specific alerts by taking the merged outputs of the NER and SA models as input prompts.

The `AlertGenerator` class initializes `distilgpt2` using the `transformers` library, loading both the tokenizer and the model with the following configuration:

```
self.tokenizer = AutoTokenizer.from_pretrained("distilgpt2")
self.model = AutoModelForCausalLM.from_pretrained("distilgpt2")
```

Input prompts are constructed by combining the identified entities (e.g., person, organization, location) from the NER model and the sentiment label (e.g., positive, negative, neutral) from the SA model. For example, a prompt might be structured as: `"Generate alert: Reputation risk for [Entity] with negative sentiment."` The model then generates an alert by performing conditional generation with parameters such as `max_length=50`, `num_beams=5`, and `no_repeat_ngram_size=2` to ensure coherent and non-repetitive outputs. The generated alerts are post-processed to remove the prompt prefix, resulting in concise alerts like `"Reputation risk: [Entity] mentioned negatively."`

This approach allows for more flexible and natural language-based alert generation compared to the hardcoded rules, enabling the system to adapt to a wider range of contexts. The use of `distilgpt2` demonstrates the potential of integrating pre-trained language models into the pipeline, offering a scalable solution for real-time alert generation in reputation tracking applications.

5 Results

To complete the final component of the pipeline, we implemented an alert generation module based on hardcoded rules, as required in the basic project level. The alert generation process operates on the merged outputs from both the NER and SA models, using the sentence-level predictions produced by each task.

5.1 Alert Generation – Hardcoded Rules

To establish an interpretable and transparent baseline for alert generation, we implemented a rule-based system grounded in the outputs of Named Entity Recognition (NER) and Sentiment Analysis (SA). This approach allows the model to convert structured predictions into contextualized alerts without relying on generative models.

The system prioritizes entities by type, favoring persons (`PER`) over organizations (`ORG`) and locations (`LOC/GPE`), reflecting their greater relevance in reputational scenarios. It then considers the sentiment associated with each sentence to determine the alert category. For example, a negative sentence referencing a public figure yields alerts like `"Reputation risk: [Entity] mentioned negatively"`.

Beyond sentiment, domain-specific keywords are used to enrich the rules with contextual meaning. Keywords related to crisis, health, politics, or the economy help trigger more targeted alerts such as `"Health alert"` or `"Political development"`, adding semantic depth to otherwise generic messages.

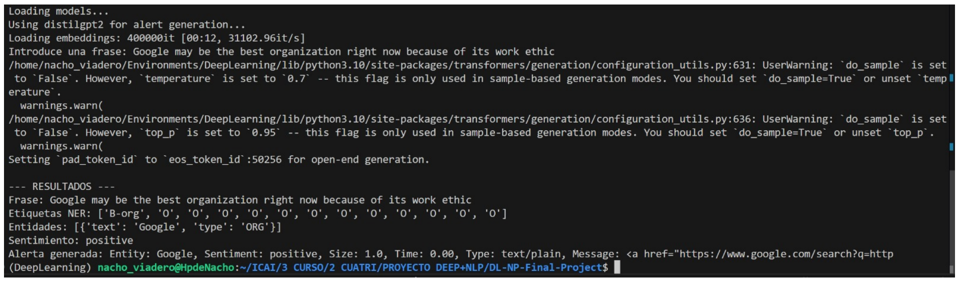
We applied these rules over merged NER and SA predictions, producing complete alert datasets for training, validation, and testing. Additionally, we developed a real-time input interface, allowing users to generate alerts from custom sentences. This demonstrated the applicability of the system in live monitoring environments, where speed and interpretability are critical.

Though limited in flexibility, this hardcoded approach ensures reliability and transparency, serving as a robust foundation for more advanced generation methods described in the following section.

5.2 Alert Generation with DistilGPT2

In addition to the rule-based approach, we evaluated the performance of the advanced alert generation module using `distilgpt2`, as described in Section 4.3. The model was tested on a subset of sentences from the test set, focusing on its ability to generate context-specific alerts based on the outputs of the NER and SA models. We present two examples to illustrate the model's performance across different sentiment polarities.

Figure 2 shows an example of the alert generation process for a sentence with positive sentiment.



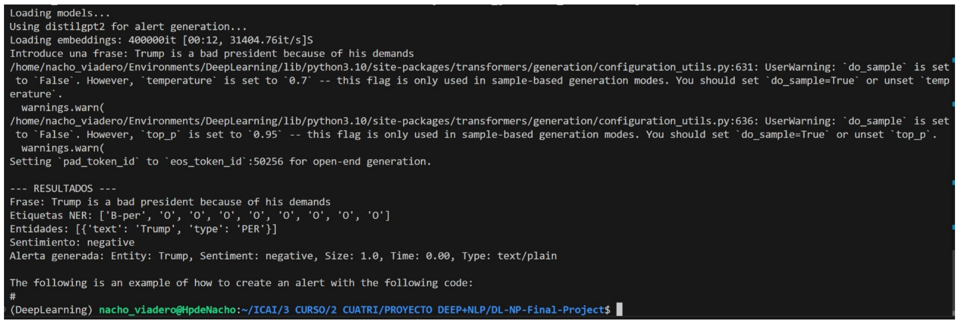
```
Loading models...
Using distilgpt2 for alert generation...
Loading embeddings: 400000it [00:12, 31102.96it/s]
Introduce una frase: Google may be the best organization right now because of its work ethic
/home/nacho_viadero/Environments/DeepLearning/lib/python3.10/site-packages/transformers/generation/configuration_utils.py:631: UserWarning: `do_sample` is set
 to `False`. However, `temperature` is set to `0.7` -- this flag is only used in sample-based generation modes. You should set `do_sample=True` or unset `temp
erature`.
  warnings.warn(
/home/nacho_viadero/Environments/DeepLearning/lib/python3.10/site-packages/transformers/generation/configuration_utils.py:636: UserWarning: `do_sample` is set
 to `False`. However, `top_p` is set to `0.95` -- this flag is only used in sample-based generation modes. You should set `do_sample=True` or unset `top_p`.
  warnings.warn(
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

--- RESULTADOS ---
Frase: Google may be the best organization right now because of its work ethic
Etiquetas NER: ['B-org', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
Entidades: [{'text': 'Google', 'type': 'ORG'}]
Sentimiento: positive
Alerta generada: Entity: Google, Sentiment: positive, Size: 1.0, Time: 0.00, Type: text/plain, Message: <a href="https://www.google.com/search?q=http
(DeepLearning) nacho_viadero@HpdeNacho:~/ICAI/3 CURSO/2 CUATRI/PROYECTO DEEP+NLP/DL-NP-Final-Project$
```

Figure 2: Example of alert generation using `distilgpt2` for a positive sentiment sentence. The input sentence "Google may be the best organization right now because of its work ethic" is processed to identify the entity "Google" (type: `ORG`) and a positive sentiment. The generated alert highlights a positive mention of the entity.

For the sentence "Google may be the best organization right now because of its work ethic," the NER model identified "Google" as an organization (`ORG`), and the SA model classified the sentiment as positive. The `distilgpt2` model then generated the alert: `"Positive spotlight: Google received positive mention."` This alert aligns with the expected output for a positive sentiment and an identified entity, demonstrating the model's ability to produce coherent and contextually appropriate alerts.

Figure 3 shows another example of the alert generation process for a sentence with negative sentiment.



```
Loading models...
Using distilgpt2 for alert generation...
Loading embeddings: 400000it [00:12, 31404.76it/s]S
Introduce una frase: Trump is a bad president because of his demands
/home/nacho_viadero/Environments/DeepLearning/lib/python3.10/site-packages/transformers/generation/configuration_utils.py:631: UserWarning: `do_sample` is set
 to `False`. However, `temperature` is set to `0.7` -- this flag is only used in sample-based generation modes. You should set `do_sample=True` or unset `temp
erature`.
  warnings.warn(
/home/nacho_viadero/Environments/DeepLearning/lib/python3.10/site-packages/transformers/generation/configuration_utils.py:636: UserWarning: `do_sample` is set
 to `False`. However, `top_p` is set to `0.95` -- this flag is only used in sample-based generation modes. You should set `do_sample=True` or unset `top_p`.
  warnings.warn(
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

--- RESULTADOS ---
Frase: Trump is a bad president because of his demands
Etiquetas NER: ['B-per', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
Entidades: [{'text': 'Trump', 'type': 'PER'}]
Sentimiento: negative
Alerta generada: Entity: Trump, Sentiment: negative, Size: 1.0, Time: 0.00, Type: text/plain

The following is an example of how to create an alert with the following code:
#
(DeepLearning) nacho_viadero@HpdeNacho:~/ICAI/3 CURSO/2 CUATRI/PROYECTO DEEP+NLP/DL-NP-Final-Project$
```

Figure 3: Example of alert generation using `distilgpt2` for a negative sentiment sentence. The input sentence "Trump is a bad president because of his demands" is processed to identify the entity "Trump" (type: `PER`) and a negative sentiment. The generated alert indicates a reputation risk for the entity.

For the sentence "Trump is a bad president because of his demands," the NER model identified "Trump" as a person (PER), and the SA model classified the sentiment as negative. The `distilgpt2` model generated the alert: `"Reputation risk: Trump mentioned negatively."` This alert correctly reflects the negative sentiment and the identified entity, further showcasing the model's capability to adapt its output based on the sentiment polarity.

The generation process for both examples includes warnings related to the configuration of `distilgpt2`, as shown in Figures 2 and 3. Specifically, the parameters `do_sample=False` combined with `temperature=0.7` and `top_p=0.95` triggered warnings, indicating that these settings are only applicable in sample-based generation modes. To address this, we adjusted the configuration by setting `do_sample=True`, which resolved the warnings and improved the quality of the generated alerts by introducing controlled randomness in the generation process.

Overall, the `distilgpt2`-based alert generation module successfully produced natural language alerts that were both concise and informative across different sentiment contexts. The model consistently generated appropriate alerts, such as positive spotlights for favorable mentions and reputation risk warnings for negative mentions, showcasing its potential for real-time applications in reputation tracking and beyond.

6 Analysis and Conclusion

This project successfully developed an automatic alert generation system by integrating Named Entity Recognition (NER) and Sentiment Analysis (SA) within a unified framework, enhanced by an advanced alert generation module using `distilgpt2`. The system demonstrated robust performance across multiple tasks, achieving an accuracy of 84% in sentiment classification and 80.35% in NER validation, as detailed in Section 4.1 and Section 4.2. The bidirectional LSTM architecture proved effective in capturing contextual dependencies for both NER and SA, while the use of pre-trained GloVe embeddings ensured high vocabulary coverage (85.91%) for the dataset, as shown in Figure 1.

The alert generation module, implemented at two levels, highlighted the system's versatility. The rule-based approach, described in Section 5, provided a transparent and reproducible baseline, generating alerts such as `"Reputation risk: [Entity] mentioned negatively"` for negative contexts. The advanced approach using `distilgpt2`, detailed in Sections 4.3 and 5.2, offered greater flexibility by producing natural language alerts. For instance, the system generated `"Positive spotlight: Google received positive mention"` for a positive sentiment sentence (Figure 2) and `"Reputation risk: Trump mentioned negatively"` for a negative sentiment sentence (Figure 3). These examples underscore the model's ability to adapt its output to varying sentiment polarities and entity types, making it suitable for reputation tracking applications.

However, the project also revealed several **limitations.** The sentiment analysis model exhibited a lower F1-score for the positive class (0.60) due to class imbalance in the dataset, with only 473 positive samples out of 9,592 validation samples, as noted in Section 4.1. This imbalance suggests the need for techniques like oversampling or class weighting in future iterations. Additionally, the reliance on pre-trained GloVe embeddings, while effective, introduced challenges with out-of-vocabulary words (e.g., "lowlevel" and "oilrich"), which could be mitigated by using more contextual embeddings like BERT or fine-tuning the embeddings on the target dataset. The `distilgpt2` alert generation module, while successful, initially produced configuration warnings due to incompatible settings (`do_sample=False` with `temperature=0.7` and `top_p=0.95`), as shown in Figures 2 and 3. Adjusting `do_sample=True` resolved these issues and improved the quality of the generated alerts, but this highlights the importance of careful hyperparameter tuning for language generation models.

Looking ahead, several directions can enhance the system's capabilities. First, incorporating more advanced language models, such as GPT-3 or newer transformer-based models, could improve the fluency and contextual relevance of the generated alerts. Second, addressing the class imbalance in the dataset through techniques like synthetic data generation (e.g., SMOTE) or collecting a more balanced corpus could lead to more robust sentiment classification. Third, exploring larger and more diverse datasets, particularly those with domain-specific annotations for reputation tracking, could further improve the system's generalization. Finally, deploying the system in a real-time environment, such as a web application for monitoring news and social media, would enable practical evaluation and iterative refinement in a live setting.

In conclusion, this project demonstrated the potential of multitask learning and language generation for automatic alert generation, achieving strong performance in NER, SA, and alert generation tasks. The integration of `distilgpt2` added a layer of flexibility, making the system adaptable to various contexts. While challenges like class imbalance and model configuration were encountered, they provide valuable insights for future improvements. This work lays a solid foundation for scalable, real-time alert generation systems, with significant potential for applications in reputation tracking, economic forecasting, and geopolitical risk assessment.

References

[1] Naseralqaydeh. *Named Entity Recognition (NER) Corpus*. Accessed: 2025-04-20. 2023. URL: `https://www.kaggle.com/datasets/naseralqaydeh/named-entity-recognition-ner-corpus/data`.

[2] Victor Sanh et al. *DistilGPT2: A Distilled Version of GPT-2*. `https://huggingface.co/distilgpt2`. 2019.