Multilingual Characterization and Extraction of Narratives from Online News via Large Language Models

Mattia Viglino
S328941

Arturo Adelfio
S316716

Vincenzo Montana
S322977

Abstract

In this paper, we present our approach for tackling the SemEval 2025 Task 10, focusing on "Multilingual Characterization and Extraction of Narratives from Online News." Specifically, we address the Entity Framing subtask (Subtask 1) on English news articles. We adopt a two-step classification pipeline leveraging `Llama 3.2 3B` and a BERT-based similarity mechanism to retrieve relevant examples. Although our final scores were lower than we had hoped, our approach offers valuable insights into the complexities of fine-grained role classification. By identifying key bottlenecks and proposing directions for improvement, we aim to promote advancements in both performance and interpretability for entity framing systems. We detail our methodology, present our findings, and discuss potential steps to further refine narrative extraction from online news. Our code implementation is available on GitHub.

1 Introduction

The internet has greatly expanded the reach of news and information, simultaneously increasing the risk of deceptive content and deliberate manipulation efforts. Large audiences can be influenced online, especially during critical events (e.g., conflicts, pandemics, climate crises), which are often accompanied by the spread of harmful disinformation and propaganda.

To encourage progress in the automatic analysis of the digital news ecosystem and facilitate the study of potential manipulation, SemEval 2025 introduced Task 10: "Multilingual Characterization and Extraction of Narratives from Online News." The main goal is to identify, classify, and extract key narrative elements in multilingual news articles.

1.1 Task Overview

The challenge comprises three subtasks across five languages: Bulgarian, English, Hindi, Portuguese, and Russian. The task covers news articles from two domains, namely, Ukraine-Russia War and Climate Change. Each subtask addresses a different aspect of narrative analysis:

- **Subtask 1: Entity Framing.** Classify each named entity mention in a news article into one or more fine-grained roles spanning three main role classes: *protagonists*, *antagonists*, and *innocent*. This is a multi-label multi-class text-span classification task.
- **Subtask 2: Narrative Classification.** Assign to each article all the appropriate subnarratives from a predefined two-level taxonomy. This is a multi-label multi-class document classification task.
- **Subtask 3: Narrative Extraction.** Generate a free-text explanation (up to 80 words) that supports the dominant narrative choice, grounded in text fragments that substantiate the narrative. This is a text-to-text generation task.

In our work, we tackle **Subtask 1** on English news articles. The dataset comprises news articles labeled with mentions of named entities and their assigned roles.

Below is a small illustrative example of the data annotation:

- `EN_UA_DEV_100012.txt Washington 1441 1450 Antagonist Corrupt`

In total, the main roles are *protagonist*, *antagonist*, and *innocent*, each subdivided into various refined roles. For example, *antagonist* may have labels such as *Instigator*, *Spy*, *Corrupt*, etc.

1.2 Data Distribution

Figure 1 illustrates the distribution of the three main roles (*antagonist*, *protagonist*, and *innocent*) in our English training dataset. As shown, the data exhibits a substantial imbalance, with *antagonist* dominating the distribution.

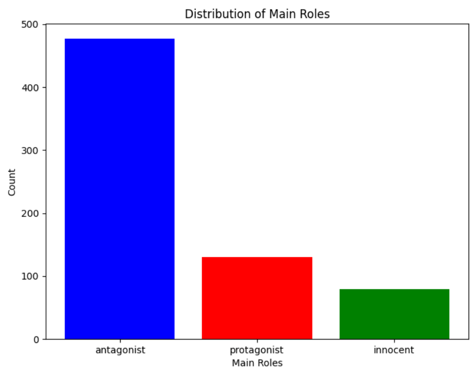

Figure 1: Distribution of Main Roles.

Table 1 breaks down the refined role distribution within each main role. For instance, within the *antagonist* category, roles such as *Deceiver*, *Corrupt*, *Conspirator*, and *Incompetent* each appear over 60 times, whereas *Spy* occurs only 3 times, revealing further data imbalance even within a single main role.

Main Role	Refined Role	Count
Antagonist	Deceiver	66
	Corrupt	63
	Terrorist	28
	Foreign Adversary	46
	Incompetent	60
	Instigator	70
	Saboteur	24
	Conspirator	79
	Spy	3
	Bigot	20
	Traitor	14
	Tyrant	53
Protagonist	Rebel	22
	Martyr	14
	Guardian	47
	Underdog	15
	Virtuous	23
	Peacemaker	18
Innocent	Victim	60
	Scapegoat	9
	Exploited	10
	Forgotten	2

Table 1: Refined role frequencies within each main role.

We will discuss that this class imbalance, present both at the main role and refined role levels, can pose complications at inference time.

1.3 Background on BERT and LLaMa

BERT is a widely adopted pretrained language model that learns deep bidirectional representations of text. Its contextual embeddings have proven effective across many downstream tasks such as classification and entity recognition. In this work, we use `BERT-base-cased` to generate semantic vectors, which help us identify the most relevant training examples for our in-context demonstration approach.

LLaMa (**L**arge **La**nguage **M**odel **A**utoregressive) is a family of transformer-based generative language models introduced by Meta. Trained on extensive corpora for next-token prediction, LLaMa models excel in text generation. In our pipeline, `Llama 3.2 3B` is deployed in a classification scenario: given a prompt with selected examples, a (development/test) text sample and an entity mention from the text, it outputs the probability for each assignable label. We employ this framework both for determining the *main* role (protagonist, antagonist, or innocent) and for further specifying the *refined* role from a set of possible sublabels.

1.4 Approach Summary

Our solution consists of a two-step classification method using the generative LLM `LLaMa 3.2 3B` to label each named entity mention:

1. A *main role* classification step that decides if an entity is *protagonist*, *antagonist*, or *innocent*.

2. A *refined role* classification step, conditioned on the predicted main role, that labels the mention with more specific categories (e.g., *Deceiver*, *Corrupt*, *Traitor*).

The two-step approach has the advantage of ensuring coherence between the main and refined role choices, as well as among the refined roles themselves. Moreover we use a BERT-based semantic similarity module to retrieve relevant training examples for in-context demonstrations during both steps. The next sections elaborate on these components.

2 Method

In this section, we detail each step of our proposed system, focusing on data preprocessing, example retrieval, and role classification.

2.1 Data Preprocessing

The organization provides the participants with two labeled datasets: training and development sets. A further (non-labeled) test dataset is provided. For subtask 1 labeled datasets consist of a set of narrative texts and an annotation document. Each annotation document record contains:

- article_id
- entity_mention
- start_offset
- end_offset
- main_role
- fine-grained_roles (one or more)

Similar dataset structure is available for test phase. In this case, however, participants must complete each annotation document record, inserting the main and fine-grained roles according to the prediction of the system they deployed.

We parse these records and read the corresponding news articles. The text of the article is then processed using a BERT-based model to obtain a semantic vector representation. This vector is used for similarity-based example retrieval. We further add special tokens within the text in order to highlight the entity to be classified.

Generating Semantic Vectors with BERT

Our classification pipeline relies on demonstrating in-context examples to the LLM, following evidence from [1] that large language models significantly improve their performance when provided with representative examples of a task. To ensure the examples are semantically relevant, we:

1. Obtain a **semantic vector** for each training article using `BERT-base-cased`.
2. Store the semantic vector as a field of the preprocessed dataset, to speed up the computations at inference time.

Special tokens insertion

We add the special tokens <T> </T> for remarking the entity to be classified. This could improve model's attention on specific parts of the text when the same entity name repeats along the narrative.

2.2 Classification via LLM

Once data have been preprocessed, we can start the inference phase, making use of `LLaMa 3.2 3B`. Here we devised the classification pipeline in a 2-step fashion: (1) main role and (2) refined role classification phases. Both of them share common approaches in building the prompt.

1. For each test/development sample, compute the cosine similarity between its vector and each training article's vector.
2. Select the top-k most similar training examples to serve as in-context demonstrations.

Main Role Classification

For *main role* prediction, we retrieve two examples for each of the three possible roles: *Protagonist*, *Antagonist*, *Innocent*. We use the generative model for classification, formulating the prompt as follows:

1. A **system context and persona** guiding the model's classification role, specifying that it must label each entity as one of *innocent*, *protagonist*, or *antagonist*.
2. An **example section**, showing the $k = 2$ retrieved training examples for each possible main role. Each example includes the full article, the entity and its corresponding main role label.
3. A **query portion** with the development/test sample. The model is provided with the narrative to be classified. It must choose one between the three possible roles.

To extract the final label, we compare the probabilities (softmax of the logits) of the starting tokens for the labels `protagonist`, `antagonist` and `innocent` and assign the main role according to the highest-probability token. We decided to use the uncapitalized version of the labels to increase the probability of the `innocent` class. In our LLaMa tokenizer, the word 'Innocent' is tokenized with 'In' as the first token, whereas the lowercase version 'innocent' starts with 'inn', which is more meaningful.

Refined Role Classification

Given the predicted main role, we next identify which refined roles apply. Each main role has an associated set of refined roles:

- `protagonist` $\rightarrow$ {*Guardian*, *Martyr*, *Peacemaker*, *Rebel*, *Underdog*, *Virtuous*}

- `antagonist` $\rightarrow$ {*Instigator*, *Conspirator*, *Tyrant*, *Foreign Adversary*, *Traitor*, *Spy*, *Saboteur*, *Corrupt*, *Incompetent*, *Terrorist*, *Deceiver*, *Bigot*}
- `innocent` $\rightarrow$ {*Forgotten*, *Victim*, *Exploited Scapegoat*}

The prompt format is similar to the main role one, but focuses on these refined roles alone. The LLM is prompted to answer with `Yes` or `No` in order to indicate if the label is appropriate for the entity. We choose all roles for which the predicted probability of `Yes` exceeds that of `No`. This approach ensures that the model evaluates each refined role independently, promoting focused attention on each possible classification. If no refined role reported higher probability for `Yes`, we select the label having the highest `Yes` token probability.

2.3 Implementation Details

The following summarizes key implementation details:

- **Libraries and Models** We use the `transformers` library for BERT, `Llama 3.2 3B` for the generative classification, and standard Python libraries for data handling and evaluation.
- **Hardware** All experiments were run on a single GPU with 16GB of VRAM. To manage GPU memory usage, we carefully discard model outputs and intermediate tensors after computing each step of the forward pass.
- **Prompt Optimization** The structure of the prompt was designed based on the performance obtained on the development set. For example we discovered that giving bigger space to the example section than the context resulted to be successful in main role accuracy increasing.
- **Code Structure** We maintain a jupyter notebook mainly structured in two phases:
  1. Data loading and preprocessing
  2. Classification phase

3 Experimental Results

We evaluate our system on the official test set of Subtask 1 (English subset). The primary metrics we focused on are:

- **Main Role Accuracy**, the fraction of named entities for which the predicted main role matches the gold label. This metric evaluates how well the model assigns the correct main role to an entity based on its context.
- **Refined Role Exact Match Ratio**, the fraction of named entities for which the predicted *set* of refined roles exactly matches the gold set. Since entities may be associated with multiple refined roles, this metric strictly measures whether the predicted roles perfectly align with the ground truth without any missing or extraneous elements.
- **Refined Role Micro Precision**, the fraction of correctly predicted refined roles out of all predicted refined roles. This measures the model's ability to avoid assigning incorrect refined roles, reflecting how precise its predictions are.
- **Refined Role Micro Recall**, the fraction of correctly predicted refined roles out of all actual refined roles. This evaluates the model's capacity to retrieve all relevant refined roles for a given entity.
- **Refined Role F1**, the harmonic mean of precision and recall for refined roles. This metric balances precision and recall, providing a comprehensive assessment of the model's refined role predictions.

The challenge ranking was based on the Refined Role Exact Match Ratio, that results to be a particularly demanding metric, ensuring that models are assessed based on their ability to assign the correct set of roles without any deviations. We can further compare our results with a baseline provided by the organization that assigns the most common role and sub-role based on training data (Table 2). We can notice that although our system under-performs in terms of Exact Match Ratio, it also reports a higher recall resulting more exhaustive in the generation of all the refined roles required.
Table 3 summarizes our results on the test set.

Discussion

These results indicate that the task of identifying and refining entity roles in the news articles remains quite challenging. The main role accuracy of 25.1% suggests that the system often struggles

Metric	Baseline Score
Main Role Accuracy	0.285
Refined Role Exact Match Ratio	0.038
Refined Role Micro Precision	0.047
Refined Role Micro Recall	0.042
Refined Role F1	0.044

Table 2: Baseline results for Subtask 1 on the test set (English).

Metric	System Score
Main Role Accuracy	0.251
Refined Role Exact Match Ratio	0.013
Refined Role Micro Precision	0.047
Refined Role Micro Recall	0.167
Refined Role F1	0.073

Table 3: System results for Subtask 1 on the test set (English).

even at a high-level classification (i.e., distinguishing *innocent*, *protagonist*, or *antagonist*). As we adopted a 2-step approach, this difficulty is further worsen when assigning fine-grained labels, since we narrow the refined roles options based on the main role assignment. Indeed the exact match rate is low (1.3%). We can moreover spot that recall is relatively higher (16.7%) compared to precision (4.7%), meaning that probably when the system generates all the correct labels, also assigns more than required. These findings highlight the complexity of capturing nuanced role distinctions and underscore the need for additional refinements in prompt engineering or changes in system design to improve performance on Subtask 1. We also add that, although our work focuses on in-context learning approaches, class imbalances (discussed in section 1) can cause issues in finding relevant examples during both the phases of our 2-step approach.

4 Conclusions

We presented a two-step pipeline that combines a generative LLM and a BERT-based in-context example retrieval for entity framing classification. Our method highlighted that, although our LLama model is able to understand the context, it struggles in tackling this classification task, probably because a bigger version of the family is needed. The refined role classification remains a challenging task, with the corresponding performance lagging behind the main role predictions. Moreover, since the possible refined roles are conditioned on the main role label predicted in the first step, any error in main role classification can propagate and further result in poor performance in refined role assignment. Nevertheless, the two-step approach promotes coherence in the model's predictions and reduces uncertainty in fine-grained role assignment, because the set of assignable refined labels is constrained by the main role predicted in the first step.

Further findings

Although we experimented with forcing specific token probabilities to guide the classification, we also conducted an additional experiment where the model generated labels more freely. This allowed us to investigate whether any intrinsic reluctance was affecting the probabilities. By removing explicit probability constraints, we could better observe whether the model was indeed holding back on attributing certain negative or politically charged roles, as might be expected from LLM safety and alignment mechanisms.

We observed that, especially in cases where a named entity was involved in a context of intense social or geopolitical conflict, the model frequently hesitated to provide a concrete refined role (e.g., labeling a political figure as *Corrupt* or *Deceiver*). In some instances, the model attempted to remain neutral or to generate cautionary disclaimers, indicating uncertainty or a deliberate avoidance of strong assertions.

While this built-in caution can be beneficial in mitigating overly risky predictions or defamatory statements, it occasionally hindered the classification performance in Subtask 1.

Future directions to improve performance include:

- Enhancing prompt engineering to better clarify the semantic nuances of each refined role and guide the model toward more consistent labeling of politically sensitive entities mitigating model biases.
- Exploring larger model architectures that potentially achieve stronger alignment with ground-truth roles.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.