

9. Detecting heroes, villains, and victims

This work was conducted in close collaboration with Diego Gómez-Zarà.

As I’ve discussed elsewhere in this thesis, by using common narrative structures and cultural references, framing enables a communication source to present and define an issue within a “field of meaning” [5]. Framing helps to contextualize events by connecting with readers’ prior knowledge, cultural narratives, and moral values [8]. It can present the agents involved in a story as heroes, villains, or victims, so that readers can more easily anticipate and comprehend the attitudes, beliefs, decisions, and actions of these agents as characters. Narrative frames are widespread throughout media, be it films, literature, or news. Their use reflects pervasive cultural models for structuring the presentation of political discourse and identities [10,12]. As a key element of popular culture storytelling, they use emotionality to unambiguously distinguish between good and evil through clear depictions of victimization, heroism, and villainy [2]. Frames tend to use positive terms to describe heroes, and negative terms for victims and villains [7,13]. Culturally, heroes embody courage, outstanding achievements, or other noble qualities, whereas villains embody evil intentions, plotting, and other negative qualities. In sum, narrative frames are critical to how readers understand new situations in terms of previous ones, and thereby make sense of the causes, events, and consequences reported by news articles [3].

We present a system that detects how the main characters of a news article are framed as heroes, victims, or villains. As currently implemented, this system interacts with news consumers directly through a browser extension. This approach can also be used to identify roles in well-understood event sequences in a more prosaic manner, e.g., for information extraction.

9.1 A model for role detection

A standard approach to recognizing narrative frames is to analyze the semantic relations among the different entities in the narrative, and with respect to the events it describes [1,11]. However, determining these relations—understanding the events in a narrative, and the roles that the entities in that narrative play in those events—is a complex, difficult, and indeed, unsolved, computational challenge. Moreover, to the extent that we do understand how to address this challenge, the techniques involved can be quite expensive computationally. Using standard coreference resolution packages, for example, our system can take more than 20 seconds to analyze a single sentence. Users simply will not wait that long to see the classification results for an entire news article.

We propose a different approach: Rather than determining the specific events and event types described in the narrative, and the semantic relations among the entities involved in those events, in detail, we suggest analyzing the entities at a much higher level of abstraction—specifically, in terms of whether they possess the qualities of heroes, victims, or villains, as conveyed by the terms used to describe them. This approach suggests a relatively simple computational realization, in which the terms closest to each entity are assessed with respect to their similarity to the terms typically

used in connection with heroes, villains, or victims. News articles, in particular, facilitate this approach given their use of consistent styles. We provide a simple, computationally cheap, and reasonably (although not perfectly) accurate role classifier.

More specifically, we seek to classify entities involved in narratives into the semantic / thematic categories “heroes,” “victims,” and “villains”—categories which are highly abstract and thus broadly applicable, and yet are also likely to be comprehensible to typical users. To accomplish this, we have developed a framework and platform for role detection that comprises the following core procedures:

1. Recognize and rank the main entities present in the news article based on prominence (see 9.1.2 for more details).
2. Identify the terms closest to these entities.
3. Use sentiment analysis to determine the polarity of these terms, and thus the sentiment ascribed to the entity.
4. Calculate the similarity of these terms to those typically used to describe heroes, villains, and victims using standard term sets or dictionaries.
5. Classify the main entities as determined in step 1, according to the proposed semantic categories, using their prominence and the analyses provided by steps 3 and 4.

9.1.1 Entity recognition

The system starts by identifying the main characters involved in a news story. Taking the news article’s headline and text as inputs, it analyzes them for relevant entities using spaCy version 2¹. The system originally used a custom NER using NLTK, but we found that performance improved significantly when we transitioned to spaCy. For each entity, the system saves the relevant terms and relations and tracks all the possible name forms to establish their connections later. For example, a person could be called “John Smith” and also be mentioned in the article as “Mr. Smith,” “Smith,” or just “John.” The system tracks all these references, and reduces and merges them in a later step. Entities must be identified as a *Person*. The system does not resolve pronoun references. Doing so might improve accuracy, but it would also add significant computational complexity, which may make the system take too long to remain useable.

¹ <https://spacy.io/>

N.R.A. Suggests Trump May Retreat From Gun Control

By MICHAEL D. SHEAR, SHERYL GAY STOLBERG and THOMAS KAPLAN MARCH 1, 2018

WASHINGTON — The top lobbyist for the National Rifle Association claimed late Thursday that President Trump had retreated from his surprising support a day earlier for gun control measures after a meeting with N.R.A. officials and Vice President Mike Pence in the Oval Office.

The lobbyist Chris Cox posted on Twitter just after 9 p.m. that he met with Mr. Trump and Mr. Pence saying that “we all want safe schools, mental health reform and to keep guns away from dangerous people. POTUS & VPOTUS support the Second Amendment, support strong due process and don’t want gun control #NRA #MAGA”

Mr. Trump tweeted about an hour later, “Good (Great) meeting in the Oval Office tonight with the NRA!”

Sarah Huckabee Sanders, the White House press secretary declined to provide details about the previously unannounced meeting. A spokeswoman for the N.R.A.’s lobbying arm, which Mr. Cox leads, did not respond to requests for further comment.

Fig. 9.1 An example of Named Entity Recognition.

headline entities and those mentioned in the news article’s body. This generates a list of several entities with their respective name forms and locations in the text. Common titles such as “Mr.”, “Mrs.”, “Ms.”, and “Miss” are ignored.

9.1.2 Prominence score

Once the system recognizes all the entities, it must detect and merge the multiple references to a given entity and select the most relevant entities in the article for role assignment. First, the system merges different references to the same entity and selects the longest entity that has proper nouns as the canonical form (e.g., the best results for “Mr. Smith,” “John Smith,” and “Smith,” would be “John Smith”). During this process, the system counts the number of times that each entity is mentioned in the text. It then calculates the relevance score of each merged entity as:

$$p_i = \alpha h_i + \alpha^{f_i} + \frac{n_i}{\|s\|}$$

Equation 9.1. p_i is the prominence score of the merged entity i , h_i is a dummy variable where 1 means that the merged entity was mentioned in the headline (0 otherwise), n_i is the number of times that the merged entity is mentioned in the body of the text, s is the total number of sentences in the text, and f_i is the first location where the merged entity is mentioned in the text.

In other words, we treat entities that (a) appear in the headline, (b) are frequently mentioned in the article, and (c) are first mentioned near the beginning of the text as more prominent. The coefficient α is to weight the impact of a mention in the news article’s headline as well as the first mention of the entity. In this implementation, $\alpha=0.4$

Finally, the system returns the three most prominent entities. We chose to return three because we are ultimately trying to match three roles.

Because of their significance, the system uses the headline first to detect the main characters of a story. Headlines present an optimal context for interpretation since the heroes, victims, or villains are often highlighted [6]. The system next analyzes the entities in the text, sentence by sentence, saving their locations in a list using the indices of their corresponding sentences. During this process, the system establishes the connection between the

9.1.3 Role dictionaries and role score

To determine the roles that these entities play in the narrative frame, the system checks the terms related to each entity and calculates their similarity with terms that typically describe heroes, villains, and victims. For each role, we created a term set or dictionary D_j , where j belongs to the set {"hero," "villain," "victim"}. The terms in each set were manually selected from news articles and blogs, as well as synonyms and antonyms. In total, each role dictionary comprises approximately 200 words.

Our intention with this step is to calculate how similar the terms related to each entity are to each role's dictionary. The system calculates the similarity of each term-pair by using WordNet, a lexical database of English [9] (contained in NLTK). Specifically, we use the built-in methods for calculating Wu-Palmer Similarity [14]. This method returns a score denoting how similar two words are, based on the depth of the two senses in the taxonomy and that of their most specific ancestor node.

We assume that the terms used to describe entities are usually closer to their respective characters in the text. Based on this assumption, the system calculates for each relevant entity its distance to other terms present in the sentence, and then prioritizes the similarity scores of the closest terms. More specifically, we define $S_i \in \{1, 2, \dots, M\}$ as the set of M sentences where the entity i is mentioned. Each element of S_i has a list of terms $\{w_{m,1}, w_{m,2}, \dots, w_{m,n}\}$, where n is the total length of the m^{th} sentence. The system calculates the similarity of a term w with respect to a role j by calculating the average of the similarities between that term w and the k most similar terms d from the dictionary D_j , as follows:

$$sim(w, j) = \frac{\sum_{d \in D_j} s(w, d)}{\|D_j\|}$$

Equation 9.2

To calculate the similarity that entity i has with role j in sentence m , we sum the similarity scores for each term in the sentence. To implement our proximity assumption, we apply a decay factor f according to the distance between the entity i and each term analyzed. Additionally, to take into account active vs. passive roles—heroes and villains are assumed to be more active, victims more passive—the system analyzes whether the entity i is the subject of the sentence (i.e., the entity that performs the action) or the object of the sentence (i.e., the entity that receives the action). In the first case, the similarity of words identified in the hero or villain dictionaries are added to the score. If the entity is the object of the sentence, the similarity of the words related to the victim dictionary are added to the score. In summary, we calculate the score of the merged entity i for role j in sentence m as:

$$r_{i,j,m} = \sum_{w \in S_k} sim(w, j) * f^{d(i,w)}$$

Equation 9.3. f is the decay factor between 0 and 1, $d(i, w)$ is the distance between the entity i and the term w of the sentence m .

Finally, the system calculates the overall role scores for each entity by averaging all the scores computed using sentences in which they occur. The results are role scores for each relevant entity of the news article's headline and body.

9.1.4 Sentiment analysis and sentiment score

The program calculates and saves the sentiment score of the terms appearing in the same sentence as the relevant entities, multiplying the sentiment by the same decay factor raised to the power of the distance as we used in the previous step ($f^{d(i,w)}$). We calculate sentiment for each term using standard toolkits, in the current implementation *VADER Sentiment*² (selected because it has the largest vocabulary of word polarities of the packages we could find), yielding a polarity for each term, ranging from -1.0 (very negative) to 1.0 (very positive). These sentiment values summed for each sentence, and then the sentiment for each sentence is averaged.

9.1.5 Role detection

In the original algorithm, the entity with the highest similarity score for the given role was selected to fill that role. This proved to be problematic, as there was no cutoff at which we said, “The score is too low, therefore there is no hero.” In the next section I describe how we resolved this issue.

9.2 Validation and automatic methods

The system as described thus far appeared to work well, but appearances can be deceiving. Therefore we created a labeled data set for validation, which also allowed us to calculate automatic role cut-offs.

9.2.1 The data

A research assistant and I each coded approximately 600 articles from the New York Times Annotated Corpus, of which 197 overlapped. A total of 908 unique articles were coded. However, 306 of these articles were excluded because they were too short, contained only stock quotes or recipes, or contained more than one story.

In each article, we selected at most one hero, one villain, and one victim. This turned out to cause some difficulty, as formulas for intercoder reliability expect that in each data instance, we are choosing between the same set of options. Not so, the way we handled it. In order to allow us to calculate intercoder reliability and later, precision and recall, we made the following approximation: We pretended that we had instead coded every entity (automatically detected with entity recognition) as hero, villain, victim, or none. Because we had only allowed one of each role per story, this led to a lot of entities being coded as ‘none’. However, having read and coded 600 of the stories, I can say with some confidence that articles where there are more than one entity that qualifies for a role are rare. Thus, I did not feel that this represented a barrier to using this approximation. The intercoder reliability was calculated using Cohen’s Kappa, which is appropriate for categorical data such as ours, and found to equal 0.447, which indicates moderate agreement.

9.2.2 Automatic role cut-off

The system as described above always returned exactly one hero, one villain, and one victim for each article – even articles for which there is no hero, or no villain, or no victim. This is clearly wrong. To address this issue, I designed a system that uses the

² <https://github.com/cjhutto/vaderSentiment>

coded data discussed in 9.2.1 to automatically generate a cutoff score for each role. For a given role, an entity cannot fulfill that role if its score is lower than the role's cutoff score.

In order to calculate the cutoff, we get the similarity scores for each entity in the training set for the role we are calculating. Then we hand the scores and their labels to a Linear Support Vector Classifier, which calculates the line that separates the two categories most effectively. Then, the cutoff value C is:

$$C = -\frac{\textit{intercept}}{\textit{coefficient}}$$

Equation 9.4

Unfortunately the results using this cutoff were not very good. I proceeded to experiment with a number of options, testing on the training set to avoid overfitting. The method I ultimately settled on differs significantly.

First of all, only entities with the correct sentiment for their role were returned. For heroes, this meant positive or neutral sentiment, and for villains and victims, it meant negative or neutral sentiment. These are sorted by their prominence score, and the three most prominent remaining entities are returned.

Next the system trains a support vector classifier using the SciKit Learn implementation of LIBSVM [4] and the 'rbf' kernel. I originally selected SVMs because I could use the line it finds to separate the two classes to calculate a cutoff value. When iteration led me to give the SVM more features and use its trained model to make predictions rather than using its coefficient and intercept to calculate a cutoff value, I experimented with the two implementations of SVMs in SciKit Learn, and also with the available kernels, and found that LIBSVM with the 'rbf' kernel yielded the best results. For features I used the similarity score, the prominence score, and the sentiment score.

9.2.3 Automatic dictionary generation

The system as described above uses ad-hoc dictionaries for each role, which were created by reading many articles and looking for words that were associated with the heroes, villains, and victims portrayed in those articles. Whenever we use an ad-hoc dictionary, there is some question as to whether a machine learning solution would yield better results. Now that I had a labeled data set, I could answer that question.

In order to do so, I begin by generating a dictionary for each entity in each article in the training set. Each word count is modified by a distance decay factor, as seen in Equation 7.3, and also multiplied by inverse document frequency.

At this point we have a functional but very skewed training set. Most of the entities are classified as "False". In order to prevent this from skewing the machine learning algorithm, I balanced the data set, including a random sample of the 'non-role entities' equal in size to the number of 'role entities' based on the hand-coded labels.

Finally, I feed the resulting labeled dictionaries into a Linear Support Vector Classifier. This allows me to use SciKit Learn's 'SelectFromModel' function, which selects the top features based on their learned coefficients; this method can only be used with some of the machine learning models in SciKit Learn, and so to some extent this did dictate my choice of algorithm. In this case I set the cutoff to $0.75 \times \text{mean coefficient value}$, and the error coefficient C to 5, after experimenting with different values for each. This yielded dictionaries of approximately 20-25 words each – much smaller than the ad-hoc dictionaries, which you may recall had approximately 200 words each.

9.2.4 Results

In 9.1.5, I outline a complex system whereby I assign roles to entities. In our validation, we used two methods for assigning roles to entities. Method 1, which I call the simple cutoff, uses the trained SVM to predict whether the returned entity with the highest similarity score fits the role. If it predicts that the entity fits the role, the system returns that entity. If not, it repeats this process for the entity with the next highest similarity score, until all three have been eliminated. If none of them fit the role according to the SVM, the system predicts that there is no entity that fills the role for that article. Because this is done independently for each role, theoretically, an entity could get assigned more than one role. Method 2, or what I called the complex cut-off, combines the complex system from 9.1.5 with the SVM to determine whether or not an entity should be assigned a role. With this method, an entity could only ever be assigned one role. It was unclear whether this difference in methods would result in an improvement in performance, so we validated using both methods.

I also calculated a baseline according to the following algorithm:

1. Take the three most prominent entities in the article, plus a blank 'entity' to represent when no entity is assigned to the role under consideration
2. Randomly select a role
3. Randomly select one of the four entities
4. Predict that entity has that role
5. Repeat until all three roles have been assigned

This is a more generous baseline than simple random chance, which would have started with three random entities from the article, rather than the three most prominent.

	Accuracy	Precision	Recall	F1
Baseline	0.264	0.122	0.201	0.151
Ad-Hoc Dictionary	0.541	0.491	0.569	0.520
Machine Learning Dictionary	0.580	0.531	0.547	0.535
AHD, complex	0.554	0.515	0.524	0.512
MLD, complex	0.593	0.578	0.518	0.543

Table 9.2.1 Validation scores for the Hero role.

A note about accuracy, precision, and recall, before I proceed with the results. Accuracy is calculated based on the way that the data was coded – zero or one hero, zero or one villain, and zero or one victim per article. Precision and recall, however, cannot be calculated with the data coded in this manner. Therefore, as with intercoder reliability, I approximated. Each automatically identified entity was treated as though it had been coded as either none, hero, villain, or victim. Again, I felt that this approximation was reasonable as cases where there is more than one of a given role in an article are rare.

	Accuracy	Precision	Recall	F1
Baseline	0.262	0.128	0.183	0.150
Ad-Hoc Dictionary	0.5	0.366	0.354	0.344
Machine Learning Dictionary	0.525	0.392	0.403	0.394
AHD, complex	0.45	0.225	0.207	0.211
MLD, complex	0.512	0.268	0.217	0.239

Table 9.2.2 Validation scores for the Villain role.

In order to avoid random chance biasing the results, I used five-fold cross-validation. As in 9.2.3, the data set was very skewed, as most instances were labeled as not having the role I was validating. Therefore, I balanced the data set using the same method outlined in 9.2.3 before proceeding with cross-validation.

	Accuracy	Precision	Recall	F1
Baseline	0.262	0.118	0.159	0.135
Ad-Hoc Dictionary	0.537	0.39	0.567	0.429
Machine Learning Dictionary	0.55	0.37	0.48	0.41
AHD, complex	0.562	0.486	0.28	0.355
MLD, complex	0.487	0.466	0.164	0.242

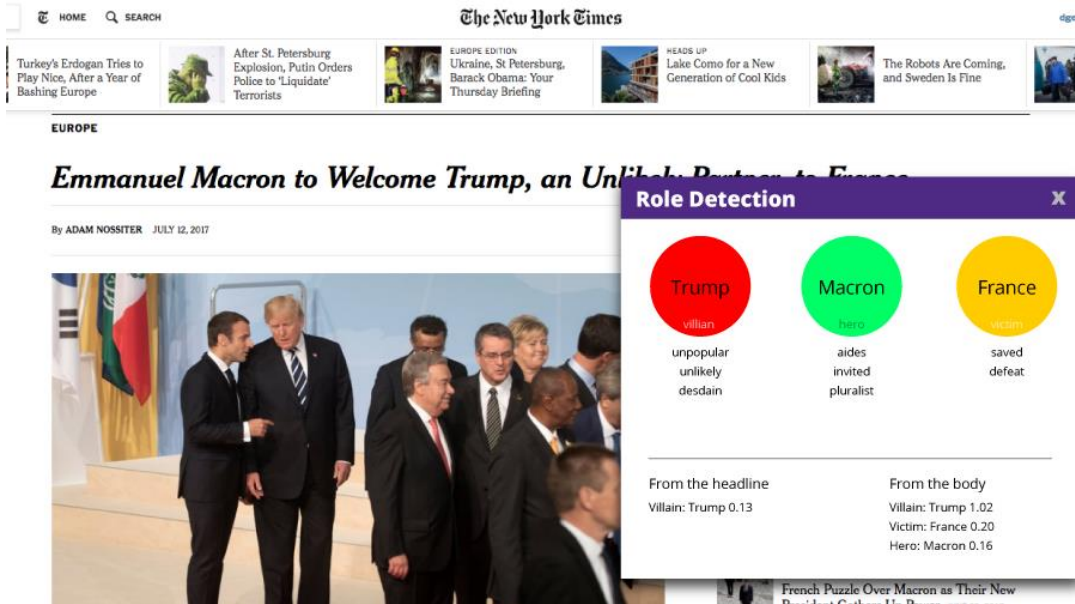
Table 9.2.1 Validation scores for the Victim role.

Overall, these results are encouraging, particularly the results for the Hero role. This difference is to be expected, as most of the algorithmic tuning occurred only with respect to the Hero role, due to time constraints. In particular, I note that all of our results outperformed the baseline by a significant margin, which itself should be better than random chance. These results are also in line with the state-of-the-art of other similarly complex NLP tasks.

Given that the Machine Learning Dictionaries were so small in comparison to the Ad-Hoc Dictionaries, I was surprised to find that the Machine Learning Dictionary outperformed the Ad-Hoc Dictionary on almost all measures for all three roles. The complex cut-off results were also a surprise; this technique was supposed to improve performance but instead the results were quite mixed. For the Hero role, the Machine Learned Dictionary with Complex Cut-off outperformed the simple cutoff Machine Learned Dictionary *and* Ad-Hoc Dictionary on all measures except recall. For the Villain role the complex cut-off generally had worse performance. And for the Victim role, the complex cut-off outperformed the simple cut-off for precision, but not accuracy, recall, or F1.

Since the complex cut-off prioritizes categorizing the entity-role pair with the highest score first, I must ask if perhaps the Hero scores tend to be higher than the Victim or Villain scores. This would explain why the Hero results generally improved with complex cut-off, but the Villain and Victim results were generally worse. If I can verify that this is a factor, I could explore normalizing score ranges so that the highest score possible for each role is the same.

9.2 Interface and examples



The screenshot shows the New York Times homepage with a headline: "Emmanuel Macron to Welcome Trump, an Unlikely Partner to France". A "Role Detection" overlay is positioned on the right, displaying the following information:

Entity	Role	Associated Terms
Trump	villain	unpopular, unlikely, disdain
Macron	hero	aides, invited, pluralist
France	victim	saved, defeat

Below the table, the system provides scores for each entity based on the headline and the body text:

Source	Trump	Macron	France
From the headline	Villain: 0.13		
From the body	Villain: 1.02	Victim: 0.20	Hero: 0.16

Fig. 9.2 The system detects President Trump as villain, President Macron as the hero, and France as the victim.



The screenshot shows the Fox News website with a headline: "Trump visits Paris without his friend, the mysterious...". A "Role Detection" overlay is positioned on the right, displaying the following information:

Entity	Role	Associated Terms
Paris	villain	ruined, terrorism, infiltrated, extremists
Trump	hero	praised, joked, speech, denounced
Jim	victim	friend, unclear

Below the table, the system provides scores for each entity based on the headline and the body text:

Source	Paris	Trump	Jim
From the headline	Victim: 0.13		
From the body	Victim: 0.10	Hero: 0.09	

Fig. 9.3 President Trump is detected as the hero, his friend Jim as the victim, and Paris as the villain.

Using the method described above with the ad-hoc dictionaries, we have developed a Google Chrome extension that determines which entities are the hero, villain, or victim, for any article being displayed. To better contextualize these results, the system also presents the most relevant terms that explain each classification, based on similarity with the assigned role's dictionary and proximity to the respective entity.

We will illustrate the system's operation using two examples that show how the tool distinguishes the coverage by two different news organizations of the same news

event. Figure 9.1 shows the system’s results given an article from The New York Times entitled “Emmanuel Macron to welcome Trump, an unlikely partner, to France” (July 12, 2017). In his visit to France, the NYT focused on President Trump’s relationship with the France’s president Emmanuel Macron. Based on the article’s body, the story’s results indicate “Macron” as the hero, “Trump” as the villain, and “France” as the victim. On the other hand, Fox News covered this same visit focusing on Trump’s friend Jim, who did not attend because—according to the president—Paris has been infiltrated by foreign extremists. This article was published the same day and entitled “Trump visits Paris without his friend, the mysterious Jim” (Figure 9.2). Initially, the system assigns Paris as the victim because it is included in the subject of the headline. Ultimately, however, it classifies Paris as the villain because is associated with the words “ruined,” “terrorism,” and “infiltrated.” Additionally, the system classifies President Trump as the hero and his friend “Jim” as the victim of this story.

These contrasting characterizations by different newspapers of the characters who took part in a single news event are an excellent illustration of how this tool can be useful. An individual who reads only one of the articles will be made aware of who the article treats as a hero, villain, or victim, allowing them to better recognize the bias encoded in the article. A more engaged individual who reads both articles will be able to compare and contrast these characterizations, not only allowing them to better recognize bias in the separate articles, but also allowing them to better recognize patterns of bias in different publications.

9.3 Conclusions

Our current implementation leverages standard entity recognition, sentiment analysis, and the use of term sets or dictionaries, which have yielded promising results. During development, we iteratively improved our original results by incorporating a variety of disambiguation techniques and putting more weight on headlines. However, because it uses simple proximity, this approach cannot handle situations in which the article presents complex grammar structures such as use of passive voice. Moreover, the system clearly requires additional work to improve the precision.

We have described an architecture for abstract or thematic role detection in news articles, instantiated in a browser extension. Rather than identifying entities in narratives as serving event- or topic-specific roles, such as buyer (or seller), inventor, thief, author, etc., we aim to identify them as serving highly abstract narrative or thematic roles, as either “hero,” “villain,” or “victim.” From a computational perspective, we believe that these abstract thematic roles may be significantly easier to identify than topic-specific roles involving more fine-grained semantic analysis. Moreover, in many instances abstract role identification could be useful in determining these more specific roles. For example, if a story is about a kidnapping, and we identify an entity in that story as the villain, then he or she is probably the kidnapper.

References

1. Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D. Manning. 2014. Combining distant and partial supervision for relation extraction. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1556–1567.
2. Elisabeth Anker. 2005. Villains, victims and heroes: Melodrama, media, and September 11. *Journal of Communication* 55, 1: 22–37.
3. Miriam L. Boon. 2017. Augmenting Media Literacy with Automatic Characterization of News along Pragmatic Dimensions. In *Companion of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, 49–52.
4. Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* 2, 3: 27.
5. Claes H. De Vreese. 2005. News framing: Theory and typology. *Information Design Journal & Document Design* 13, 1.
6. Daniel Dor. 2003. On newspaper headlines as relevance optimizers. *Journal of Pragmatics* 35, 5: 695–721.
7. Jeff Goodwin, James M. Jasper, and Francesca Polletta. 2009. *Passionate politics: Emotions and social movements*. University of Chicago Press.
8. Melanie C. Green. 2004. Transportation into narrative worlds: The role of prior knowledge and perceived realism. *Discourse processes* 38, 2: 247–266.
9. George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM* 38, 11: 39–41.
10. Zhongdang Pan and Gerald M. Kosicki. 1993. Framing analysis: An approach to news discourse. *Political Communication* 10, 1: 55–75.
<https://doi.org/10.1080/10584609.1993.9962963>
11. Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 492–501.
12. Dietram A. Scheufele. 1999. Framing as a theory of media effects. *Journal of communication* 49, 1: 103–122.
13. Stefan Stieglitz and Linh Dang-Xuan. 2013. Emotions and information diffusion in social media—sentiment of microblogs and sharing behavior. *Journal of management information systems* 29, 4: 217–248.
14. Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, 133–138.