(Revised) IUI Comps Proposal: Hero, Villain, or Victim? Role detection in news articles

Tianna Avery, Quinn Mayville, Yingying Wang, Megan Zhao

Vision

Deliver a Google Chrome extension that can analyze the entities (individuals, organizations, countries, etc.) contained within a news article and categorize the most important entities as either being a hero, villain, or victim.

Detailed Description

The majority of our project will entail implementing the role detection tool closely following the process outlined in the article by Gomez-Zara, Boon, and Birnbaum [1]. This tool will be packaged as a browser extension for Google Chrome. The process of creating this tool can be roughly split into three broad steps: entity recognition and ranking, role assignment, and creating an interface that interacts with the user and webpage.

Entity Recognition and Ranking

The first algorithmic step in creating the role detection tool is recognizing the entities present in the news article. We will use entity recognition methods provided by the Natural Language Toolkit (NLTK), a suite of libraries for natural language processing in Python, to extract the entities from the headline and text of the news article. The program only considers the NLTK categories of Persons, Organizations, Geopolitical Entities, or Positions (e.g. "the witness") as possibilities for heroes, villains, and victims. The program will generate a list of entities with their name forms and locations in the text (headline or index of a sentence).

Once the entities are recognized, the program needs to select the most significant entities from the article for role assignment. To do this, it will first merge different references to the same entity, using the longest name associated with proper nouns as the final form of the entity. We still need to figure out how to implement the merging method for references. We plan to learn the technique while proceeding with other aspects of our project. For each merged entity, a relevance score r is calculated as $r = \alpha h + \frac{n}{|s|f}$, where α is a scaling parameter, h is a dummy variable signaling if the entity was in the headline, n is the number of times the entity is mentioned in the text, $|s|$ is the total number of sentences in the text, and f is the first location where the entity is mentioned in the text. The program then narrows down the entities to include

all entities mentioned in the headline and the top three entities from the text with the highest relevance scores.

Role Assignment

Before assigning roles, we will assemble a dictionary for each role that contains terms that typically describe heroes, villains, and victims, respectively. We will either construct these dictionaries manually from news articles and thesauruses, or we will get the original dictionaries from the researchers. To assign the roles, the program will measure the similarity of terms used to describe entities with the words in the role dictionaries. In order to avoid detecting the high similarity of antonyms, we will use TextBlob, a Python library that can perform sentiment analysis, to calculate the sentiment of each term. Positive terms will be compared to the hero dictionary, while negative terms will be compared to the villain and victim dictionaries.

The actual role detection algorithm considers all sentences in which the entity being analyzed is mentioned. For each term w in the sentence S, the similarity of the term to role j is computed as follows:

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

where M_j is the dictionary of role j and $s(w, m)$ is the Wu-Palmer Similarity between words w and m. Wu-Palmer Similarity is a method built into the NLTK that returns a score denoting how similar two meanings of a word are, based on the depth of the two meanings in the taxonomy and that of their Least Common Subsumer (most specific ancestor node) [3]. The score of an entity i for role j in sentence S is then computed by:

$$score(i,j,S) = \sum_{w \in S} \{[sim(w,j) + a(i,j,w)](1-f)^{d(i,w)}\}$$

where $a(i, j, w)$ is an additional score according to the positions of the term w and entity i in the sentence, f is a decay factor, and $d(i, w)$ is the distance between i and w in the sentence. Using this additional score, if an entity has an active role in the sentence the hero and villain scores will increase, while an entity with a passive role will have an increase in the victim score. Further research is needed to determine exactly how the additional score should be calculated. In the equation above, $[sim(w,j) + a(i,j,w)](1-f)^{d(i,w)}$ is an exponential decay function, structured as $y = a(1-b)^x$. Following this, $[sim(w,j) + a(i,j,w)]$ is the similarity score before the decay occurs and f is the percentage by which the similarity score will decline as the distance between i and w increases. The model will average the scores across the sentences in which they occur,

resulting in role scores for all relevant entities. The program then assigns the entity with the highest role score to the corresponding role and continues to do so until all roles are assigned.

Interface

We will need to construct a Chrome extension to extract the news article data from the webpage and to run and display the role assignment tool. Creating a Chrome extension is a straightforward task, as the extension only consists of an HTML page with some CSS and JavaScript. We will need a compiler to convert our Python code to JavaScript, but there are plenty of tutorials on how to do this. RapydScript is a public pre-compiler that best fit our purpose. RapydScript has a unique syntax that is slightly dissimilar from both Python and JavaScript, but this syntax is well-documented and should not be overly complicated to learn. We have successfully implemented a simple Chrome extension using Python and RapydScript, which gives us the confidence to believe that it is feasible to implement the whole extension using Python.

An essential task that our Chrome extension must perform is extracting the headline and text body from the news webpage the user is on. Our extension can interact with the webpage through JavaScript, and we plan on using a Python library to extract the headline and text (python-goose and news-please are leading possibilities). Apart from this piece, most of our work on the Chrome extension side will be designing a UI for the role detection tool.

Once we complete the program as described in the paper, we will expand upon and clean up the existing project. One clear way to expand on the project is to include a role score threshold so that the model doesn't assign roles when the role doesn't exist. For instance, many articles may not have a villain. We will also need to spend time planning a user study and considering performance indicators for the program. Finally, it is likely that much of our time improving the tool will be spent upgrading the UI.

Literature Review

In noticing a problem of readers' analysis of news articles from various news sources, Gomez-Zara, Boon, and Birnbaum [1] created a program that reviews different entities in a given news article and assigns them the role of hero, villain, or victim. The goal of this program is to "assist readers in applying their media literacy skills" [1]. Our deliverable will be primarily modeled after what this team at Northwestern University has achieved as well as what they see as possible improvements of this project. Specifically, we chose to adopt the idea to sort people into the abstract groups of hero, villain, or victim. We will adopt (with some slight manipulation) the steps in their algorithm and the equations they provide for determining the relevance score of

entities, similarities between entity terms and role terms, and that for calculating the score of entities for a specified role.

Bergstrand and Jasper [2] have published a paper recently about the significance of implicitly characterizing entities as either heroes, victims, or villains in western media. The paper talks about character theory alongside how certain characterizations may be more likely to activate the public's pathos, be more likely to motivate them into action. This article and those like it become relevant as the conversation shifts to be about the real-world implications of the project we're working on and its possible extensions.

As no one in our group has any previous experience with natural language processing, the literature on this topic will be particularly useful. One book that is cited in "Who is the Hero, the Villain, and the Victim?" [1] is "Natural language processing with Python" [3]. This book provides an introduction to natural language processing with the aid of the NLTK — a Python library that we will be using to build this project. The practical nature of this book (what Bird, Klein, and Loper describe as a balance between theory and application) will make this book perfectly suited for our intended purposes.

The documentation for other tools that we envision ourselves using to build this Chrome extension will also be important. WordNet [4], a lexical database of English, was used for some of the calculations in the original paper that we plan on adopting into our own. This project will rely heavily on the NLTK; thus, the documentation from that will be immensely helpful for us [2]. Moreover, TextBlob [5], a library built on NLTK, with its documentation will be a useful resource as it was used in the original paper to calculate sentiment of words. Finally, the official documentation on how to create a Google Chrome extension will be useful to us as we implement this concept.

Roadmap

1. Chrome extension skeleton and basic interface / Learning NLP **(10/10 - 10/25, 15 days)**
2. Extract news articles' headlines and text from websites as inputs **(10/15 - 10/22, 7 days)**
3. Entity recognition (Identify main characters involved in the story) **(10/10 - 11/15, 35 days)**
    a. Generate a list of entities with respective name forms and locations in the text **(10/10 - 11/1, 22 days)**
        i. Recognize entities using NLTK
        ii. Save entities' locations using sentence indices
    b. Merge references/different name forms **(11/2 - 11/19, 17 days)**
        i. Research how to implement the merging

    ii. Merge to the longest entity and count the number of times the entity is mentioned
    iii. Calculate the relevance score
    iv. Rank: all entities from the headline and three entities with the highest relevance score
4. Role dictionaries **(no timeline — 3 days if manually; not sure if asking authors for their dictionary)**
    a. Create 3 dictionaries for each role including synonyms and antonyms (approximately 200 words each)
5. Role comparison analysis **(10/28 - 11/15, 19 days)**
    a. A set of the *k* sentences in which an entity appears where each element is a list of all the sentence's terms **(10/28 - 11/2, 6 days)**
    b. Term priority calculation (based on the distance the term is from the entity in the sentence) **(11/3 - 11/5, 3 days)**
    c. Calculate and save the similarity score for each term (calculate the average of the similarities between that term and each term from a role's dictionary) - WordNet in NLTK or TextBlob? **(11/6 - 11/10, 5 days)**
    d. Sum up the similarity scores to calculate the similarity of each entity with a role and average all the scores **(11/11 - 11/15, 5 days)**
6. Role assignment (basically sort with assigned weights) **(1/7 - 1/20, 14 days)**
    a. Research about the additional score function **(1/7 - 1/13, 7 days)**
    b. Calculate the score of the merged entity for each role in each sentence where the entity appears **(1/14 - 1/20, 7 days)**
7. Extension and interface **(1/21 - 2/15, 24 days/depends on the UI)**

Divide and Conquer

We plan to create two sub-teams that work on separate tasks. The two subteams are Quinn and Megan (Team QM) and Tianna and Yingying (Team TY). In the initial phase, Team QM will work on entity recognition to identify the main characters involved in the news articles. Team TY will learn how to build a primary Chrome extension along with basic interfaces. Team TY will also extract news articles' headlines and text from websites as input. If time allows, Team TY will look into JavaScript and brainstorm potential UI interfaces for the final deliverable.

In the second phase, Team QM will finish up entity recognition. Team TY will start role comparison and role assignment. Team QM will assist Team TY if entity recognition related knowledge is involved. During the final phase, Tianna, Quinn, and Yingying will finish up the tasks related to role assignment. Megan and Yingying will work on the UI interface.

Our comps members have decided to work individually as well as in subteams. The subteams hope to work frequently in the same room for clear communications. Both subteams will explain what they have done during the whole group's meet.

Individual Tasks

We will each individually implement one simple function prior to the end of the fall.

- **Quinn**: Implement sim(*w, d*) function that computes the Wu-Palmer similarity between words *w* and *d* using the NLTK
- **Yingying**: Implement sim(*w, j*) function that computes the similarity between word *w* and role *j* using the sim(*w, d)* function and the role dictionary for *j*.
- **Tianna**: Implement sentiment(*w)* function that computes the sentiment of word *w* using the TextBlob library.
- **Megan**: Implement relevance score function that computes the relevance of an entity given data of the entity's occurrences in the text (not sure exactly what this input form will be yet).

Estimate of Software, Hardware, etc. (gray: optional)

- Programming Languages
    - HTML
    - CSS
    - JavaScript
    - Python
- API/Libraries
    - Natural Language Toolkit
    - TextBlob
    - Python Goose (for extracting main content from HTML)
    - Beautifulsoup
    - news-please (for extracting main content from HTML)
    - Beautiful Soup (for extracting main content from HTML)
    - newspaper (for extracting main content from HTML)
    - boilerplate (for extracting main content from HTML)
    - RapydScript (for compiling Python to JavaScript)
- Browser
    - Google Chrome

Final deliverables

The basic final deliverable of this project will be a Chrome extension that identifies the hero, the villain, and the victim for the news article being displayed. Our interface will pop up at the right bottom of the Chrome webpage. Role detection would be the title of our interface with three differently colored circles containing the entities recognized as the hero, the villain, and the victim respectively. We will also present the most relevant terms for each classification and hopefully a performance indicator to show evaluation based on user inputs or comparisons with the annotated ground truth.

In addition to the implementation of our interface in a web browser, as extensions we plan to integrate our project to social media to analyze the hero, villain, and victim of shared articles on Facebook. We would also like to update the words that we found associated with heroes, villains, and victims that are not contained in their respective dictionaries.

If time allows, we will connect with other groups at Carleton College. We could implement our features to the “Burst Your Bubble” comps group along with our goal to integrate with social media. We could also analyze the roles in the creative writing papers of English majors at Carleton. We hope to integrate data visualizations of our interface to show an average ranking of an entity from other articles. For example, if Trump is marked as a hero in one paper, but on an average across news articles, he is labeled more of a villain than hero. We could show a visualization of how he is categorized in the percentage of the three categories across the news articles.

Works Cited

[1] D. Gomez-Zara, M. Boon, and L. Birnbaum, "Who is the Hero, the Villain, and the Victim? Detection of Roles in News Articles using Natural Language Techniques," in *IUI '18: 23rd International Conference on Intelligent User Interfaces, March 7-11, 2018, Tokyo, Japan* [Online]. Available: https://doi.org/10.1145/3172944.3172993. [Accessed October 9, 2018].

[2] K. Bergstrand and J. M. Jasper, "Villains, Victims, and Heroes in Character Theory and Affect Control Theory," *Social Psychology Quarterly*, vol. 81, no. 3, pp. 228-247, August 2018. [Online]. Available: https://doi.org/10.1177/0190272518781050. [Accessed October 9, 2018].

[3] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc., 2009. [Online]. Available: https://www.nltk.org/book/.

[4] Princeton University, "WordNet: A Lexical Database for English," *WordNet*. Princeton University, 2010. [Online]. Available: https://wordnet.princeton.edu. [Accessed: October 9, 2018].

[5] S. Loria, "TextBlob: Simplified Text Processing," *TextBlob.* Steven Loria, 2013. [Online]. Available: https://textblob.readthedocs.io/en/dev/index.html. [Accessed: October 9, 2018].