FrameAnnotator - A Web-Based Frame Semantic Annotation Tool

By

Sarbajit Roy

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

Supervisor: Dr. Chengkai Li

May 2019

Abstract

News headlines around the world are alarming. They aim to trigger emotional responses from the consumer. In an era where news and social media posts spread so quickly, it is often difficult to distinguish between what is *real* and what is *false*. Human fact-checking fails to cope with the growth of such unprecedented information, thus increasing the demand and various steps of automatic fact-checking to judge the veracity of claims. An automatic fact-checking system is a statistical model that can help to detect misinformation. The current state-of-the-art frame semantic parsers suffer from lack of a large annotated data-set and there are limited annotation tools available. So, we introduce how Frame Annotator, a web-based frame semantic annotation tool, uses natural language processing and the concept of frame semantics to help users to generate annotated data-sets.

Table of Contents

FrameAnnotator - A Web-Based Frame Semantic Annotation Tool

Sarbajit Roy
Department of Computer Science and Engineering
The University of Texas at Arlington
sarbajit.roy@mavs.uta.edu

Supervisor: Dr. Chengkai Li
Committee Members: Dr. Ramez Elmasri and Dr. Deokgun Park
May 2019

Abstract

News headlines around the world are alarming. They aim to trigger emotional responses from the consumer. In an era where news and social media posts spread so quickly, it is often difficult to distinguish between what is *real* and what is *false*. Human fact-checking fails to cope with the growth of such unprecedented information, thus increasing the demand and various steps of automatic fact-checking to judge the veracity of claims. An automatic fact-checking system is a statistical model that can help to detect misinformation. The current state-of-the-art frame semantic parsers suffer from lack of a large annotated data-set and there are limited annotation tools available. So, we introduce how FrameAnnotator [1], a web-based frame semantic annotation tool, uses natural language processing and the concept of frame semantics to help users to generate annotated data-sets.

1 Introduction

In this paper we introduce FrameAnnotator, a web-based frame semantic annotation tool. The focus here is to explain the working model of FrameAnnotator, how the concept of frame semantics[2] is implemented in FrameAnnotator, the system architecture, and the purpose of FrameAnnotator.

There is a prolonged battle with the information disseminated underneath the guise of news, such as exaggerated statements and half-truth claims, where politicians constantly make identical false claims, the effects of which are horrific. These negatively affect the economy, government, health, and even security of a nation. To fight against the proliferation of false information there is an increase of fact-checking organizations. Fact-checking is the task of analyzing whether or not claims that are made are genuine or not. However, human fact-checking is laborious, time-consuming and intellectually challenging. Hence it leads to the need for *automated* fact-checking systems (Babakar and Moy, 2016; Hassan et al., 2017a,b; Graves, 2018). The most important pre-requisite for training any fact-checking system is the availability of a diverse set of data that have been annotated (or labeled). Currently, the annotated data set is limited and to train such a system we need enough large volumes of annotated data to learn complex patterns and then integrate into a predictive model. Labeling such data is not an easy task, it requires domain knowledge, researching and figuring out proof, know-how the context of information and reasoning about what can be inferred from the proof.

To address such a challenge, we developed a web-based public frame semantic annotation tool called FrameAnnotator. The user can take the help of FrameAnnotator to annotate training data-sets which can be used in diverse programs including information extraction, machine translation, sentiment analysis, among other different things. The FrameAnnotator consists of four main parts (shown in Figure 2) : (1) the input part, (2) the frame selection part, (3) the annotation part, and (4) the output part. Each element has its own significant purpose. The

[1] https://idir.uta.edu/frameannotator/
[2] https://en.wikipedia.org/wiki/Frame_semantics_(linguistics)

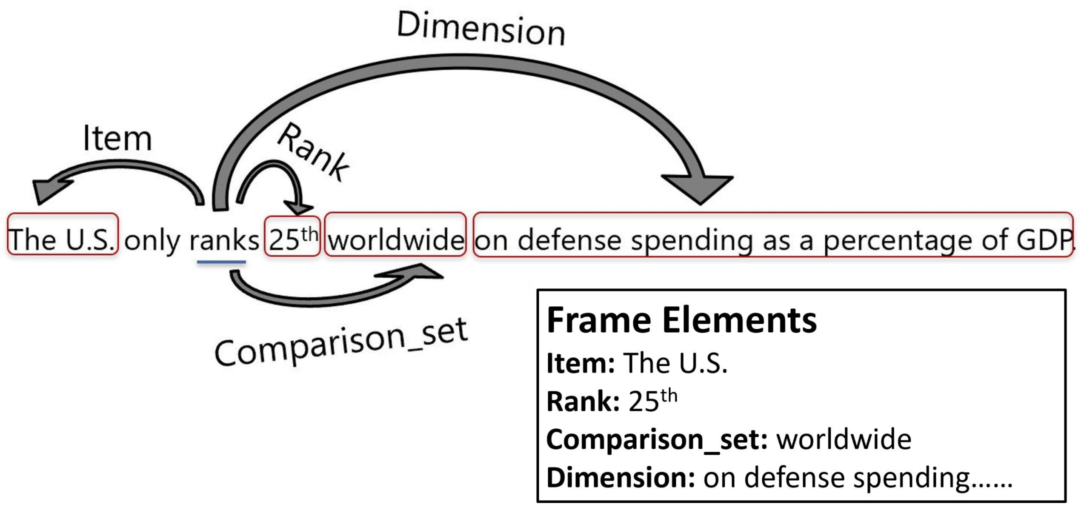

Dimension
Item
Rank
The U.S. only ranks 25th worldwide on defense spending as a percentage of GDP.
Comparison_set
Frame Elements
Item: The U.S.
Rank: 25th
Comparison_set: worldwide
Dimension: on defense spending……

Figure 1: Working of Frame-based Semantic Role Labeling

FrameAnnotator is a high-level system which supports and helps the user to annotate data efficiently and promptly.

2 Background

2.1 Frame-based Semantic Role Labeling

Semantic role modeling/labeling (Jurafsky and Martin, 2018) also know as semantic parsing, is the procedure of assigning labels to words or phrases in a sentence denoting their semantic function in that sentence. For example, given a sentence like "The U.S. only ranks 25^{th} worldwide on defense spending as a percentage of GDP.", the task is to recognize the verb "ranks" which is the *predicate*, "The U.S." as representing the *agent* (Ranking entity), "on defense spending as a percentage of GDP" as representing the *dimension* (rank based on), "Worldwide" as representing the *comparison_set* (rank comparison) and "25^{th}" as representing the *rank* (item occupies).

Figure 1 shows the working of frame-based semantic role labeling. This is an important step towards understanding the relationship of the arguments with the predicate (in this case, "ranks"). We can extract more information from the text for a higher meaning of the sentence semantic. This modeling capability is useful for a variety of fact-checking steps.

2.2 FrameNet

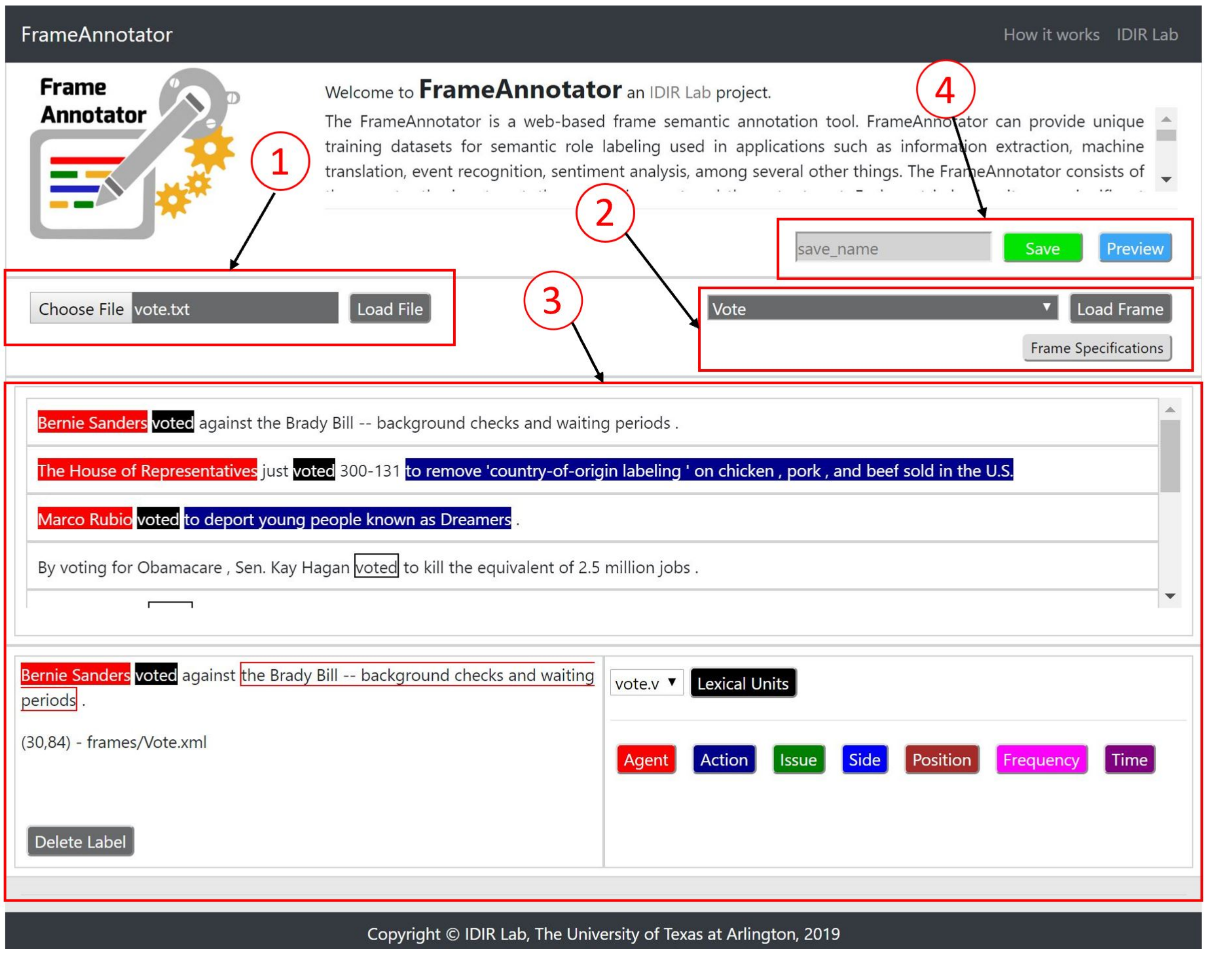
FrameNet (Baker et al., 1998) is a rich knowledge base that contains information about words by providing their description and associated frames. FrameNet is based on a theory called frame semantics.

In FrameNet [3], each frame is comprised of several components: frame definition, associated frame elements, lexical units, exemplified and annotated sentences, and frame-to-frame relations. The FrameNet database currently includes 1224 semantic frames, 13,640 lexical devices, and 202,000 annotated sentences [4].

The frame is a schematic representation of a situation which involves various participants and other conceptual roles. Frames are defined with respect to some background information which is widespread in artificial intelligence and cognitive science. Considering the words 'stand', 'top', and 'rank', we can notice the common and holistic background knowledge that unites them together. For example, the words 'stand', 'top', and 'rank' are from 'occupy_rank' Frame. Occupy_rank (Arslan et al., 2019) frame is about items in the state of occupying a certain rank within a hierarchy.

Frame elements provide additional infor-

[3] en.wikipedia.org/wiki/FrameNet
[4] https://en.wikipedia.org/wiki/FrameNet


FrameAnnotator

How it works IDIR Lab

Frame Annotator

Welcome to **FrameAnnotator** an IDIR Lab project.

The FrameAnnotator is a web-based frame semantic annotation tool. FrameAnnotator can provide unique training datasets for semantic role labeling used in applications such as information extraction, machine translation, event recognition, sentiment analysis, among several other things. The FrameAnnotator consists of

1 2 3 4

save_name | Save | Preview

Choose File vote.txt | Load File

Vote | Load Frame | Frame Specifications

Bernie Sanders voted against the Brady Bill -- background checks and waiting periods .

The House of Representatives just voted 300-131 to remove 'country-of-origin labeling ' on chicken , pork , and beef sold in the U.S.

Marco Rubio voted to deport young people known as Dreamers .

By voting for Obamacare , Sen. Kay Hagan voted to kill the equivalent of 2.5 million jobs .

Bernie Sanders voted against the Brady Bill -- background checks and waiting periods .

(30,84) - frames/Vote.xml

Delete Label

vote.v | Lexical Units

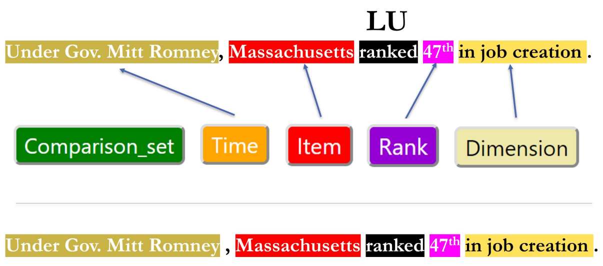
Agent | Action | Issue | Side | Position | Frequency | Time

Copyright © IDIR Lab, The University of Texas at Arlington, 2019

Figure 2: The FrameAnnotator - Web-based frame semantic annotation tool

mation to the semantic structure of a sentence. The Frame elements deliver the essential meaning and general description of the sentence.

The lexical units are the citation form of a set of words, with their part of speech, that call a specific frame. Lexical units can be associated with its specific frame(s). For each frame, there are many Lexical Units associated to one frame and many frames that share multiple Lexical Units. Frames are related to example sentences and frame elements are marked within the sentences. Thus the sentence: *"Under Gov. Mitt Romney, Massachusetts ranked* 47^{th} *in job creation."* is associated with the Occupy_rank frame, while "Massachusetts" is marked as the frame element item, "47^{th}" is marked as Rank, "in job creation" is marked as the dimension, and "Under Gov. Mitt Romney" is marked as the time. See Figure 3 to understand Occupy_rank frame. FrameNet has many limitations, the FrameNet has a very small sized corpus and users cannot add new corpus in FrameNet. The number of frames in FrameNet are fixed, that is, users cannot add new type of frames in FrameNet.



LU
Under Gov. Mitt Romney, Massachusetts ranked 47th in job creation.
Comparison_set
Time
Item
Rank
Dimension
Under Gov. Mitt Romney, Massachusetts ranked 47th in job creation.

Figure 3: Understanding an example with Occupy_rank frame

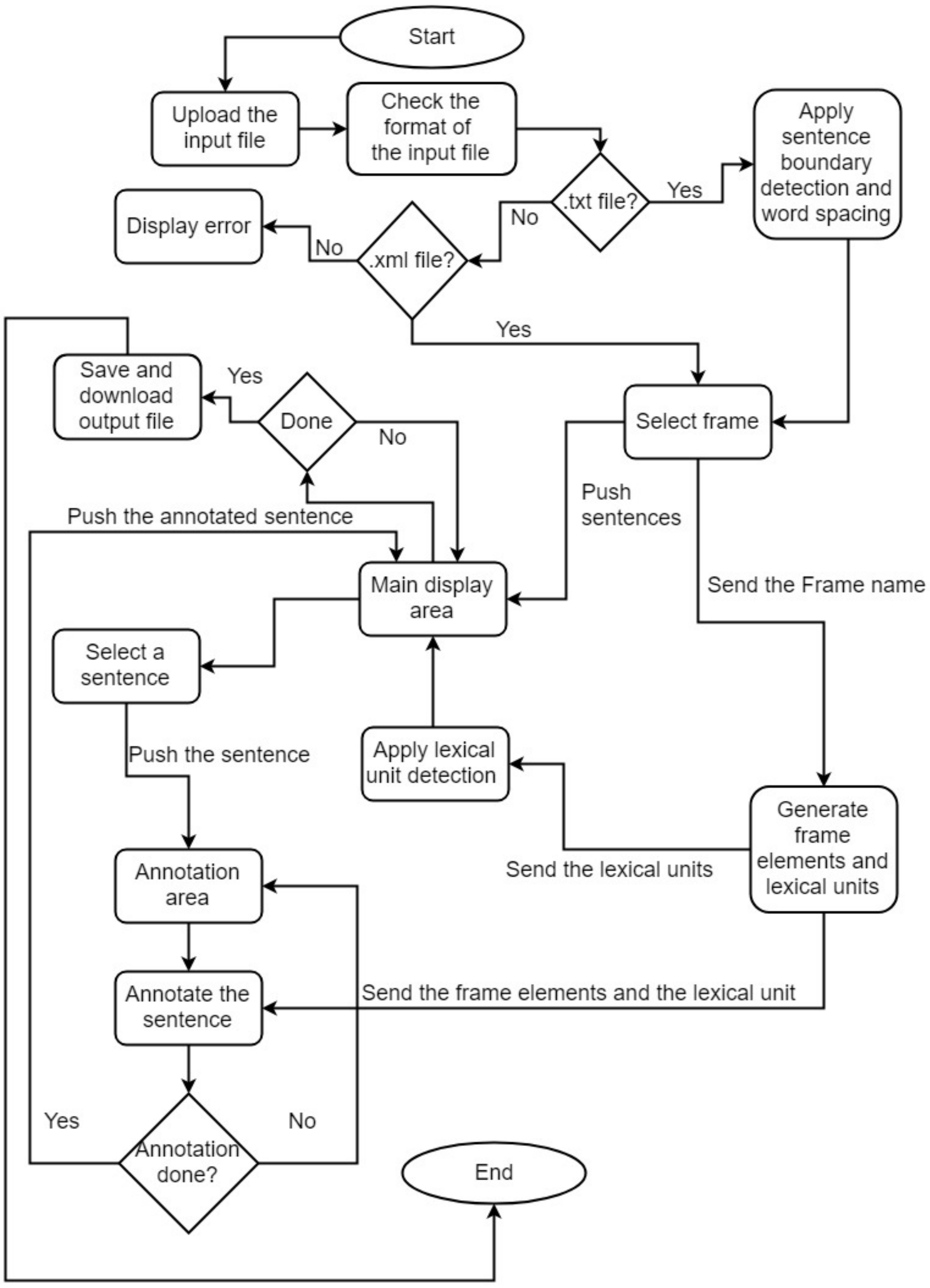


Figure 4: Flow diagram of FrameAnnotator

2.3 Factual-Claim Specific Frames

In IDIR Lab [5], they analyzed many fact-checked claims from PolitiFact [6]. They examined a subset of these claims one by one and categorized the similar ones. After this process they created new frames, that were not there in FrameNet collection. They listed 20 frames including 7 existing and 13 newly created. Few examples of factual-claim specific frames (Arslan et al., 2019) are Oppose_and_support_consistency, Vote, Correlation, Occupy_rank, etc.

[5]https://idir.uta.edu/

[6]https://www.politifact.com/

3 FrameAnnotator

The FrameAnnotator is a web-based frame semantic annotation tool Figure 2. It allows users to create frame semantic data-sets and codify sentences, based on annotating examples of how words are used in actual texts. In machine learning or deep learning, computers can use this annotated data to learn the recognition of similar patterns when presented with new data. FrameAnnotator can help to create such training data-sets for semantic role labeling used in applications such as information extraction, machine translation, event recognition, sentiment analysis, among other things.

The FrameAnnotator is a high level annotating system which supports and helps in generating annotated data-sets efficiently and promptly. The FrameAnnotator consists of four parts- the input part, the frame selection part, the annotating part and the output part. Each part has its own significant purpose. The input part (numbered (1) in Figure 2) takes the input file, pre-processes it and sends it to the annotating part (numbered (3) in Figure 2). The frame selection part (numbered (2) in Figure 2) helps the user to select the frame. The automatic components take care of the sentence boundary detection, the word-spacing (separating words from special characters) in the sentence, generates frame element buttons and lexical unit button, and lexical unit detection. The annotating part takes care of the entire annotating mechanism. The output part (numbered (4) in Figure 2) saves the entire annotation work and returns the result in a pre-defined readable format. We can refer Figure 4 to understand the flowchart representation of FrameAnnotator. The working model of FrameAnnotator follows:

1. The user **import a .txt file or an .xml file** to annotate the data.

2. After importing the file, the user needs to **select the frame from the frame drop-down list**.

3. The user can **view the frame specification** by clicking the frame specification button to refer the selected frame. Frame specification contains all frame related information with annotated examples.

4. With the selection of the frame, the **system automatically creates all the frame element buttons**. The system has the capability of detecting the lexical unit(s) in the input data. The user can take the help of the frame element buttons to annotate the data.

5. The user can use the **delete label button to remove the annotation** from the word or phrase.

6. The user can **save the annotated work by clicking the save button** and it will **automatically download the output file**. The user can also use the **preview button to view the output on the web browser**.

7. The system has the facility to help the user to **resume the work** just by importing the saved output file in the input part.

3.1 System Architecture

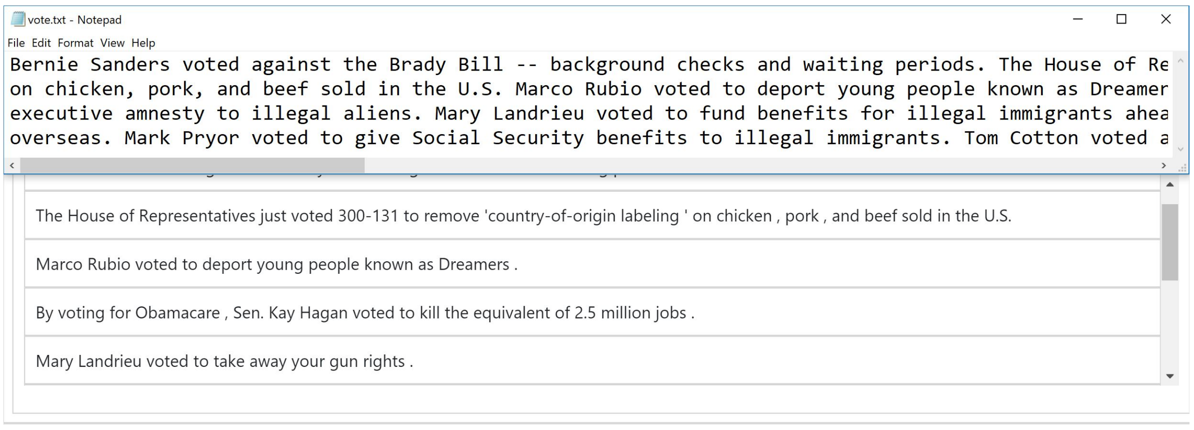
The FrameAnnotator consists of front-end and back-end components. The front-end is a user-friendly representation of functionality that a user interacts with. It does not need any specific OS/device-related adjustments. The user interface of FrameAnnotator is developed in HTML [7], JavaScript [8] and CSS [9]. The back-end is the main control center of FrameAnnotator with which the user can interact. The back-end framework of FrameAnnotator is scripted using PHP [10] language. JavaScript is used to implement the functionalities - the import of data from a .txt file or an .xml file, the listing of sentences from the import data, the selection of an individual sentence from the list for annotation, the word or phrase selection for annotation, fetching the data from frame .xml file to create frame element buttons and lexical unit drop-down list, implementing the delete label button functionality, sentence boundary detection, word spacing and parts of speech tagging. The PHP scripting language is used to

[7] https://en.wikipedia.org/wiki/HTML

[8] https://en.wikipedia.org/wiki/JavaScript

[9] https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[10] https://en.wikipedia.org/wiki/PHP



```
Bernie Sanders voted against the Brady Bill -- background checks and waiting periods. The House of Re
on chicken, pork, and beef sold in the U.S. Marco Rubio voted to deport young people known as Dreamer
executive amnesty to illegal aliens. Mary Landrieu voted to fund benefits for illegal immigrants ahea
overseas. Mark Pryor voted to give Social Security benefits to illegal immigrants. Tom Cotton voted a
```

The House of Representatives just voted 300-131 to remove 'country-of-origin labeling ' on chicken , pork , and beef sold in the U.S.

Marco Rubio voted to deport young people known as Dreamers .

By voting for Obamacare , Sen. Kay Hagan voted to kill the equivalent of 2.5 million jobs .

Mary Landrieu voted to take away your gun rights .

Figure 5: Listing of sentences from raw text

incorporate the conversion of annotated and non-annotated data into a .xml file. The .xml file format is used to create common information format that is both human-readable and machine-readable. For parts of speech tagging and sentence boundary detection we used wink [11]. Wink is an open source packages for Statistical Analysis, Natural Language Processing, and Machine Learning in NodeJS.

3.2 Importing Data files

The system takes the input as a .txt or a .xml file. If the file is a .txt file then the system calls the sentence boundary detection function and word spacing function before displaying the entire text in the annotation part. The sentence boundary function automatically divides the entire text into list sentences and the word spacing function helps to separate words from special characters in the sentence. If the file is a .xml file then the system extracts all the sentences with their respective annotated features (to show the annotated words or phrases) for the .xml file and display them to the annotation part.

Input file format can be of two types: a **.txt** file or a predefined **.xml** file. A .txt file consists of raw non-annotated texts. A .xml file consists of annotated sentences with their respective attributes and also non-annotated sentences. The user can save their work and resume it later by uploading the .xml file. The format of the .xml file is pre-defined in the system. The system will be unable to read the .xml file if the format of the file is incorrect.

Sentence boundary detection helps to detect the sentence boundaries [12] in the input file and split the entire text into the list of sentences. Our assumption here is that each sentence will have only one frame and one lexical unit associated with it. Word spacing helps to separate words from special characters by using blank-space so that to consider all words and special characters individually. This helps in part of speech tagging and it is important because the meaning of text generally depends on the relationship between words in that text. We can refer to Figure 5 to understand the working of sentence boundary detection and word spacing with an example.

3.3 Frame Selection

The system provides the user with a frame drop-down list to select the frame. The frame is a .xml file that consists of frame definition, frame ID, frame elements along with their IDs, font and background colors, the frame relation, and lexical units with the parts of speech associated with it. On selecting the frame from the drop-down list, the system automatically fetches all the related information from the .xml file of the selected frame to create frame element buttons, to help in detecting lexical unit(s) in sentences, to create lexical unit button and lexical unit drop-down list. Currently, the FrameAnnotator has 20

[11] https://www.npmjs.com/package/wink-pos-tagger

[12] https://winkjs.org/wink-nlp-utils/

Frames, listed in Table 1.

Frame Names
Capability
Causation
Change_position_on_a_scale
Cause_change_of_position_on_a_scale
Comparing_two_entities
Comparing_at_two_different_points_in_time
Conditional_event_occurrence
Correlation
Creating
Oppose_and_Support_Consistency
Occupy_rank
Occupy_rank_via_ordinal_numbers
Occupy_rank_via_superlatives, Ratio
Recurring_action
Recurrent_action_in_frequency
Speech
Taking_sides
Uniqueness_of_trait
Vote

Table 1: Type of Frames

The frame specification is an accordion button which will help the user to refer the selected frame. The frame specification contains all frame related information such as, frame definition, type of frame elements with their description, the lexical units and example of annotated sentences. The system automatically loads and displays the frame specification section after selecting the frame. We can refer Figure 6 to see a pictorial representation of Frame specification accordion button.

Frame elements are created for the frame chosen by the user. The system automatically creates frame element buttons with their respective font and background color. The system directly fetches all the related information such as, the frame element name, font color, background color, frame ID, and frame element ID. For an example if we see Vote frame, the frame elements of Vote frame are ”Agent”, ”Action”, ”Issue”, ”Side”, ”Position”, ”Frequency”, and ”Time”. We can refer Figure 8 to see the frame elements of Vote frame.

The lexical unit button, and lexical unit drop-down list are created automatically above the frame element buttons after the frame is selected. The lexical unit button will help us to annotate the lexical word in the sentence.

The lexical unit detection function associates with part of speech tagging helps to detect the lexical unit in the sentence. Based on the selected frame, it takes the lexical unit name(s) from the selected frame .xml file and compare it with the all the words in the sentence to highlight the matching words. The part of speech helps the system to find the predicate in the sentence. In Figure 7 we can see a graphical representation of automatic lexical unit detection.

3.4 System Annotation Mechanism

The FrameAnnotator takes individual sentences at a time for annotation. We can refer Figure 9 to see the annotation steps.

The display section in FrameAnnotator is the section where all the sentences are listed. The User needs to *select one sentence at a time to annotate*. The display area shows all the annotated and non-annotated sentences.

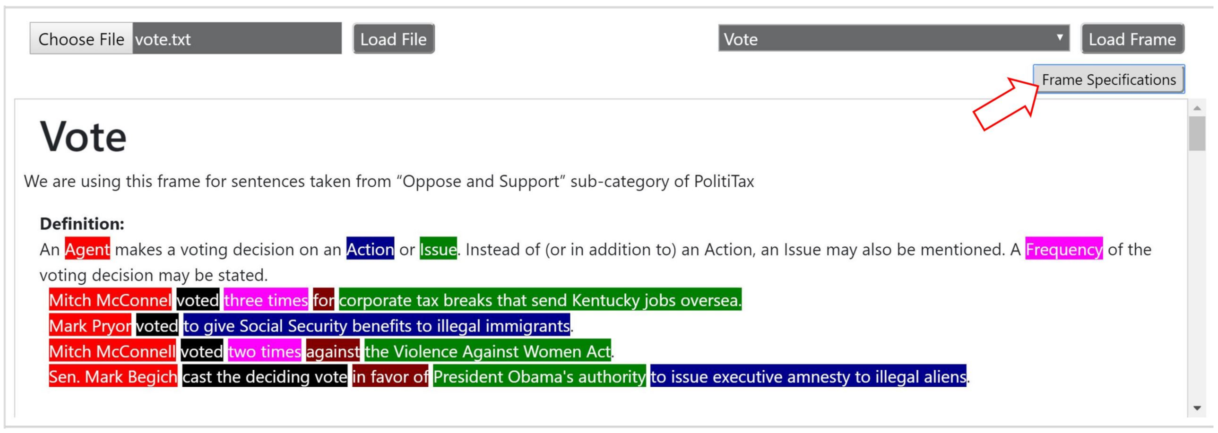
The annotation section in FrameAnnotator is the section where users annotate the selected sentence from the display section. The system is capable of selecting a word or even a phrase to annotate. User needs to click on the word to select that word and click-hold-drag to select a phrase. Delete Label button is used to delete the annotation of a word, phrase or a lexical unit.

Parts of speech (PoS) tagger labels words into categories to identify the word's function in a sentence. The common parts-of-speech categories include noun, verb, article, adjective, etc. The part of speech helps to identify the lexical units in the sentence and understand the semantic. For parts of speech tagging and sentence boundary detection, we are using Wink. Wink is a family of open source packages for Statistical Analysis, Natural Language Processing and Machine Learning in NodeJS [13].

3.5 Output Part

The output part takes care of saving the annotating part in a predefined .xml format.

[13]https://nodejs.org/en/



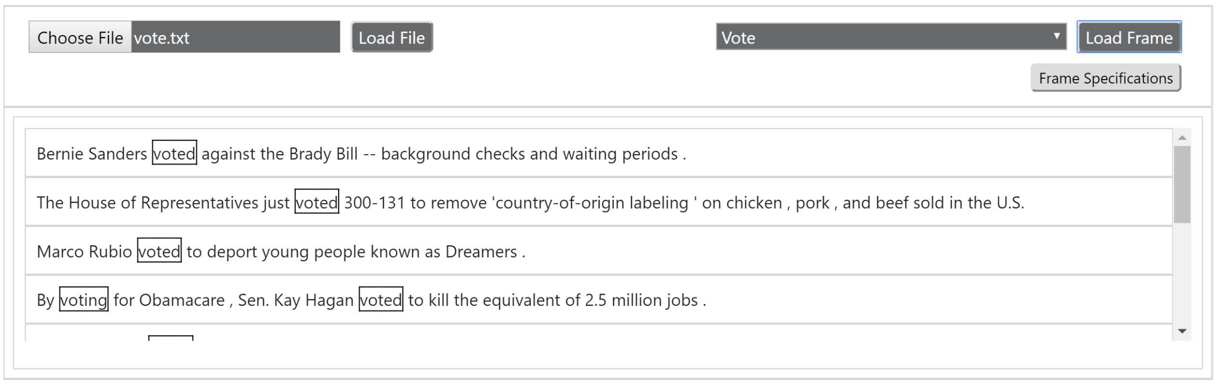
Vote

We are using this frame for sentences taken from "Oppose and Support" sub-category of PolitiTax

Definition:
An Agent makes a voting decision on an Action or Issue. Instead of (or in addition to) an Action, an Issue may also be mentioned. A Frequency of the voting decision may be stated.

Mitch McConnel voted three times for corporate tax breaks that send Kentucky jobs oversea.
Mark Pryor voted to give Social Security benefits to illegal immigrants.
Mitch McConnell voted two times against the Violence Against Women Act.
Sen. Mark Begich cast the deciding vote in favor of President Obama's authority to issue executive amnesty to illegal aliens.

Figure 6: Vote frame specification



Bernie Sanders voted against the Brady Bill -- background checks and waiting periods .
The House of Representatives just voted 300-131 to remove 'country-of-origin labeling ' on chicken , pork , and beef sold in the U.S.
Marco Rubio voted to deport young people known as Dreamers .
By voting for Obamacare , Sen. Kay Hagan voted to kill the equivalent of 2.5 million jobs .

Figure 7: Automatic Lexical Unit Detection

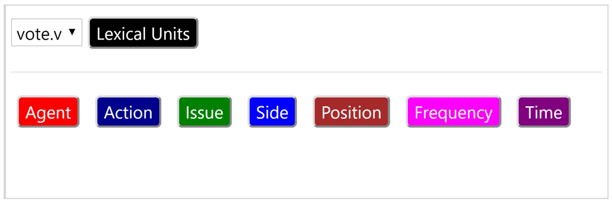


Figure 8: Frame Elements of vote

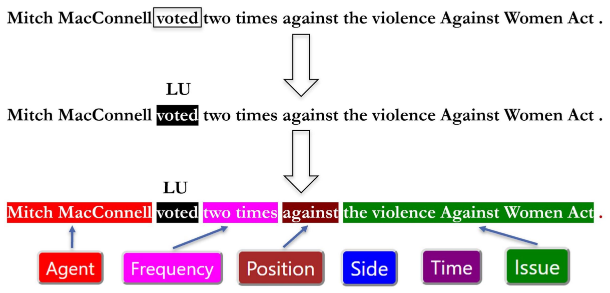

Mitch MacConnell voted two times against the violence Against Women Act .
LU
Mitch MacConnell voted two times against the violence Against Women Act .
LU
Mitch MacConnell voted two times against the violence Against Women Act .
Agent
Frequency
Position
Side
Time
Issue

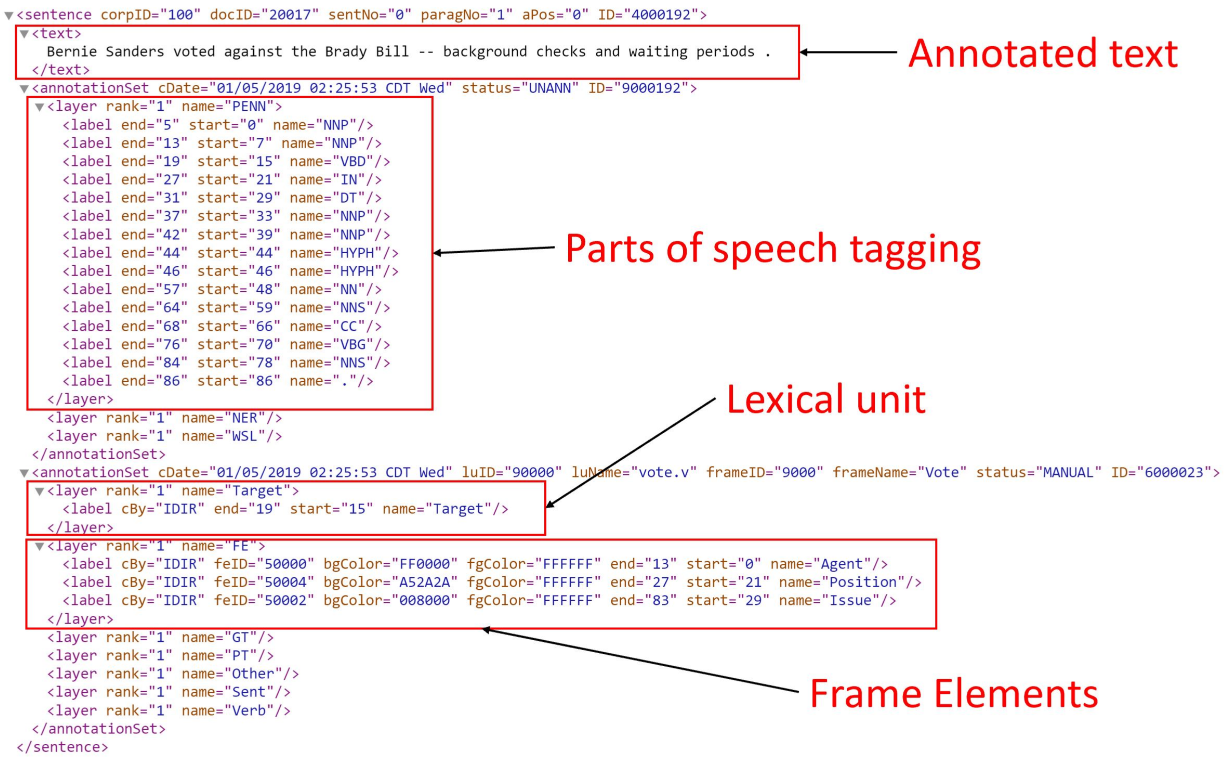
Figure 9: Annotation example using FrameAnnotator

This .xml file comprises of individual annotated and non-annotated sentences with their parts of speech, the start and end index of the words, the font and background colors of the annotated words or phrases associated with the frame elements, and many more.

To save the work the user needs to provide a file name. The system automatically saves the output file and downloads locally when the save button is clicked. The user can also preview the output file by clicking the preview button and it will open in a web browser to display the output file.

Output file format defines a set of layers for encoding documents in a .xml format that is both human-readable and machine-readable. The layers of the output .xml file:

- **fullTextAnnotation** - This is the root layer. FullTextAnnotation consists of the header, corpus, document, sentence, text, and annotationSet with all the hyperlinks
- **header** - This layer consists of the corpus layer and document layer



```
▼<sentence corpID="100" docID="20017" sentNo="0" paragNo="1" aPos="0" ID="4000192">
 ▼<text>
    Bernie Sanders voted against the Brady Bill -- background checks and waiting periods .
  </text>
 ▼<annotationSet cDate="01/05/2019 02:25:53 CDT Wed" status="UNANN" ID="9000192">
   ▼<layer rank="1" name="PENN">
      <label end="5" start="0" name="NNP"/>
      <label end="13" start="7" name="NNP"/>
      <label end="19" start="15" name="VBD"/>
      <label end="27" start="21" name="IN"/>
      <label end="31" start="29" name="DT"/>
      <label end="37" start="33" name="NNP"/>
      <label end="42" start="39" name="NNP"/>
      <label end="44" start="44" name="HYPH"/>
      <label end="46" start="46" name="HYPH"/>
      <label end="57" start="48" name="NN"/>
      <label end="64" start="59" name="NNS"/>
      <label end="68" start="66" name="CC"/>
      <label end="76" start="70" name="VBG"/>
      <label end="84" start="78" name="NNS"/>
      <label end="86" start="86" name="."/>
    </layer>
    <layer rank="1" name="NER"/>
    <layer rank="1" name="WSL"/>
  </annotationSet>
 ▼<annotationSet cDate="01/05/2019 02:25:53 CDT Wed" luID="90000" luName="vote.v" frameID="9000" frameName="Vote" status="MANUAL" ID="6000023">
   ▼<layer rank="1" name="Target">
      <label cBy="IDIR" end="19" start="15" name="Target"/>
    </layer>
   ▼<layer rank="1" name="FE">
      <label cBy="IDIR" feID="50000" bgColor="FF0000" fgColor="FFFFFF" end="13" start="0" name="Agent"/>
      <label cBy="IDIR" feID="50004" bgColor="A52A2A" fgColor="FFFFFF" end="27" start="21" name="Position"/>
      <label cBy="IDIR" feID="50002" bgColor="008000" fgColor="FFFFFF" end="83" start="29" name="Issue"/>
    </layer>
    <layer rank="1" name="GT"/>
    <layer rank="1" name="PT"/>
    <layer rank="1" name="Other"/>
    <layer rank="1" name="Sent"/>
    <layer rank="1" name="Verb"/>
  </annotationSet>
</sentence>
```

Figure 10: The output file

- **corpus** - This layer consists of corpus description, corpus ID, corpus name
- **document** - document name, document description, document ID
- **sentence** - corpus ID, document ID, sentence No., paragraph No., sentence ID
- **text** - the annotated or non-annotated sentence
- **annotationSet** - status = ”UNANN”, date, annotationSet ID, POS tags
- **annotationSet** - status = ”MANUAL”, date, lexical unit ID, luxical unit name, frame ID, frame name, annotationSet ID

4 Conclusion

In summary, FrameAnnotator is a high-level web-based frame semantic annotation system which supports and helps the user to annotate data efficiently and promptly. These annotated data can be used in diverse programs including information extraction, machine translation, sentiment analysis, among other different things. Most importantly, FrameAnnotator will help to annotate factual claims so that it can leverage the capability in serving a variety of steps for automatic fact-checking.

Acknowledgments

I would first like to thank my advisor Dr. Chengkai Li, your advice on both research as well as on my career have been invaluable. I would like to thank my committee members Dr. Ramez Elmasri and Dr. Deokgun Park for taking the time to review my thesis. I would like to thank Fatma Arslan for helping whenever I ran into a trouble spot or had a question about my research or writing.

Finally, I must express my very profound gratitude to my mother Gita Roy, my aunts Anima Dutta, Anita Dutta and Champa Paul, my elder brother Satadru Roy and sister-in-law Swati Roy, my IDIR lab-mates, all my friends, UTA CoLA and UTA Cricket team for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Last but not the least I would really like to dedicate my master's thesis to my father Shyamal Roy. Thank you.

References

Fatma Arslan, Damian Jimenez, Josue Caraballo, Gensheng Zhang, and Chengkai Li. 2019. Modeling factual claims by frames. In *Proceedings of the 2019 Computation+Journalism Symposium.*

Mevan Babakar and Will Moy. 2016. The state of automated factchecking. *Full Fact 28.*

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1. Association for Computational Linguistics.*

Lucas Graves. 2018. Factsheet: Understanding the promise and limits of automated fact-checking. Technical report, Tech. Rep.). Reuters Institute for the Study of Journalism, University of Oxford.

Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017a. Toward automated fact-checking: Detecting check-worthy factual claims by ClaimBuster. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1803–1812. ACM.

Naeemul Hassan, Gensheng Zhang, Fatma Arslan, Josue Caraballo, Damian Jimenez, Siddhant Gawsane, Shohedul Hasan, Minumol Joseph, Aaditya Kulkarni, Anil Kumar Nayak, Vikas Sable, Chengkai Li, and Mark Tremayne. 2017b. ClaimBuster: The first-ever end-to-end fact-checking system. *Proceedings of the VLDB Endowment*, 10(12):1945–1948.

Daniel Jurafsky and James H. Martin. 2018. Chapter 18 -semantic role labeling.