Summer 6-2019

A DATA DRIVEN APPROACH TO IDENTIFY JOURNALISTIC 5WS FROM TEXT DOCUMENTS

Venkata Krishna Mohan Sunkara
University of Nebraska - Lincoln, sunkara.km96@gmail.com

A DATA DRIVEN APPROACH TO IDENTIFY JOURNALISTIC 5WS FROM TEXT DOCUMENTS

by

Venkata Krishna Mohan Sunkara

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Professors Ashok Samal and Leen-Kiat Soh

Lincoln, Nebraska

June, 2019

A DATA DRIVEN APPROACH TO IDENTIFY JOURNALISTIC 5WS FROM TEXT DOCUMENTS

Venkata Krishna Mohan Sunkara, MS

University of Nebraska, 2019

Adviser: Ashok Samal and Leen-Kiat Soh

Textual understanding is the process of automatically extracting accurate high-quality information from text. The amount of textual data available from different sources such as news, blogs and social media is growing exponentially. These data encode significant latent information which if extracted accurately can be valuable in a variety of applications such as medical report analyses, news understanding and societal studies. Natural language processing techniques are often employed to develop customized algorithms to extract such latent information from text.

Journalistic 5Ws refer to the basic information in news articles that describes an event and include *where, when, who, what* and *why*. Extracting them accurately may facilitate better understanding of many social processes including social unrest, human rights violations, propaganda spread, and population migration. Furthermore, the 5Ws information can be combined with socio-economic and demographic data to analyze state and trajectory of these processes.

In this thesis, a data driven pipeline has been developed to extract the 5Ws from text using syntactic and semantic cues in the text. First, a classifier is developed to identify articles specifically related to social unrest. The classifier has been trained with a dataset of over 80K news articles. We then use NLP algorithms to generate a set of candidates for the 5Ws. Then, a series of algorithms to extract the 5Ws are developed. These algorithms based on heuristics leverage specific words and parts-of-speech customized for individual Ws to compute their scores. The heuristics are based on the syntactic structure of the document as well as syntactic and semantic representations of individual words and sentences. These scores are then combined and ranked to obtain the best answers to Journalistic 5Ws. The classification accuracy of the algorithms is validated using a manually annotated dataset of news articles.

ACKNOWLEDGMENTS

Firstly, I would like to extend my heartfelt gratitude to my advisor and mentor, Dr. Ashok Samal. His wisdom, support and guidance were instrumental in helping me successfully complete my research. I am thankful for all the weekly progress meetings he scheduled to help me in planning my thesis work. Without his guidance and inspiration this thesis would not have been possible. I deeply appreciate all the efforts he has put into helping me.

I am also grateful to my co-advisor Dr. Leen-kiat Soh for his time and patience in providing valuable feedback throughout my research. His suggestions have been invaluable throughout the course of this research and have helped it reach a successful completion.

I also would like to extend a warm thank you to Dr. Deepti Joshi, whose feedback at every point served as a stepping stone for making continuous improvement and thus enhancing the quality of this research. I would also like to thank all the undergraduate students at the Citadel who helped me in preparing the ground truth dataset.

I would like to use this opportunity to thank my committee member Dr. Jitender Deogun who took time out of his busy schedule and invested it in reviewing my work. He provided constructive opinions and also helped to add on to how this research can be further developed and improvised.

And lastly, I would like to thank my parents and sister Krishna Priya Sunkara for their unconditional love and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The world has been a conglomeration of different people who interact with each other on daily basis as a society using various resources. Understanding the growth and change in a society is essential in making rapid decisions. With the rapid growth of the Internet and connectivity, data about human interactions is now being available in various forms such as structured data in the form of text from newspapers, and unstructured data in the form of tweets, posts from social media. This large amount of data can be used for analyzing a social process and for extracting information, which help in making quick decisions to emerging changes. This analysis can also be useful for capturing the dynamics of a society that are connected across geographical regions and time.

A group of people might show their dissatisfaction against a particular policy, idea or movement by either public demonstrations or disorder, which is termed as a social unrest event. These types of unrest events can usually be found in newspapers or social media and be considered as data. Understanding different pieces of information from this data can provide valuable insight into the different factors responsible for an unrest event and can also help in taking preventative measures. However, understanding textual data requires human expertise and isn't feasible on large datasets. By the advent of natural language processing (NLP) techniques, machines can now begin to extract useful information and help us understand the data more effectively and efficiently. This research focuses on extracting useful information from textual data with the use of various natural language processing techniques.

1.1 Overview of the 5Ws Analysis

Understanding textual data is a fundamental task for a variety of use cases. For instance, Google News tries to cluster similar articles based on information in the news articles. News summarization condenses news articles by incorporating only the important information in it. Other disciplines also make use of information extraction for analyzing textual data, e.g., medical reports are analyzed to obtain information pertaining to a condition, social scientists analyze news articles to understand how media report on certain events [104]. Information extraction techniques are also used to obtain related scientific research articles in digital libraries.

Any event which is reported in news tries to answer the Journalistic 5Ws which are: *where*, *when*, *who*, *what* and *why*. The answers to these 5Ws provide most of the details required to understand an event and also provide meaningful insights to an event. This research focuses on extracting the answers to these Journalistic 5Ws using syntactical and semantic relationships within the text.

A data driven pipeline is developed in this approach to extract 5Ws candidates corresponding to an event reported in text. Several heuristics based on the syntactic structure as well as the semantic context of text documents are used for scoring the extracted candidates. Extraction of these 5Ws can be extended to incorporate different sources. The extracted results can be used for assessing the similarity among events, analyzing the evolution of an event over time or grouping related events to predict future events.

1.2 Research Challenges

There are several challenges associated with the analysis of unrest data. Data related to unrest events can be obtained from different sources but, it might contain unstructured data, irrelevant and duplicate events. The syntactic structure of text can vary across geographical regions and news sources. Understanding text might require the semantic context of words which is difficult while analyzing text from various languages. Also, it isn't guaranteed that each of the articles being analyzed has all the 5Ws listed in them or may even contain multiple 5Ws corresponding to more than one event. Development of 5Ws extraction algorithm (5WE) has many challenges as listed below:

- ***Data collection:*** News articles data related to unrest events is not readily available and depends on the credibility of existing data sources like GDELT [68], ICEWS which contains many false positives [115]. Downloading textual data of these events might be not feasible in some cases because of bad url provided and network connectivity issues.
- ***Lack of ground truth:*** No standard dataset with annotations for 5Ws exists which poses a significant challenge to the evaluation of developed techniques.
- ***NLP dependency:*** Many of the extraction algorithms depends on existing NLP algorithms which are not perfect for all tasks. These algorithms depend on the linguistic features of data, which differ for different languages. For example, verb coming before noun phrases in some languages poses a significant challenge in analyzing text.

1.3 Problem Definition

The aim of this thesis is to build a generalized pipeline which extracts the Journalistic 5Ws from an article. Given a document D corresponding to a topic T, the pipeline decides whether D belongs to topic T and then extract the $5Ws$ related to an event mentioned in D. Current approaches do not have a standard dataset for evaluating and focus only on certain event descriptors in the text. In response to this problem, we develop an automated extraction pipeline to extract latent information and evaluate them on a manually annotated dataset.

This thesis focuses on articles related to social unrest and customize the generalized approach to the social unrest domain.

1.4 Approach

We first evaluate a diverse set of classification algorithms on unrest articles and describe their results. We then describe the 5WE for extracting 5Ws from an unrest article by leveraging the syntactic and semantic cues within an article. This is achieved by making use of existing NLP techniques like parts-of-speech tagging, named entity recognition and co-reference resolution. 5WE also considers a list of topic/domain-specific words which help in identifying the 5Ws pertaining to an unrest event. Lastly, we evaluate 5WE on a manually annotated dataset.

1.5 Contributions

The contributions of this thesis are:

- A systematic evaluation of a diverse set of document classification algorithms to determine the classification accuracy on articles related to social unrest.

- An algorithm to obtain Journalistic 5Ws from text (5WE) leveraging a set of domain-specific keywords.
- Evaluation of the 5WE using a manually annotated 5Ws dataset of social unrest articles.

5WE can be easily extended to other domains and languages by using appropriate NLP models.

1.6 Thesis Outline

This Section describes the organization of the remainder of this thesis. In Chapter 2, we provide background about the various natural language processing techniques used in this thesis and discuss the related work in the field of document classification and 5Ws extraction. Chapter 3 provides a formal outline about the framework used in this thesis and then we describe the document classification techniques used and also describe in detail about 5WE. Chapter 4 describes the datasets used in this thesis and then analyzes the results obtained on these datasets. Chapter 5 concludes the report and suggests potential ideas for future work.

Chapter 2

Related Work

The first step for extracting 5Ws from text is to decide whether it is related to a specific domain or not using document classification algorithms. Section 2.1 describes various document classification algorithms. The 5WE utilizes the syntactic structure of a sentence and an entire document and other linguistic information related to it. Section 2.2 describes the natural language processing techniques used in our approach. Finally, Section 2.3 presents previous work in the field of information extraction, 5Ws extraction and causal relation ship extraction.

2.1 Document Classification

The amount of text that in the digital world is ever increasing and several automatic information extraction techniques are widely used to obtain important information from these structured and unstructured texts. Unstructured data is easily processed and understood by humans, but automatic processing requires advanced techniques that effectively analyze the text in regard to context.

Text classification has been widely studied in various communities such as data mining, machine learning and information retrieval. It is also used in various domains ranging from image processing, medical diagnosis to information retrieval, page ranking. Text classification classifies documents into a predefined set of classes by learning contextual representations from textual data. The problem of classification is defined as follows. We have a training set $D = \{d_1, d_2, d_3, ..., d_n\}$ of documents, such that each document d_i is labeled with a label l_i from a set $L = \{l_1, l_2, l_3, ..., l_k\}$. The task is to find a classification model f, which can assign correct label to a new document d (test instance).

2.1.1 Machine Learning Techniques

Machine learning is a branch of artificial intelligence that uses a set of hypotheses to find patterns in data. These patterns are then used to predict future patterns. It is widely divided into three approaches namely, Supervised learning, Unsupervised learning and Semi-supervised learning. Supervised learning methods tries to infer a function or learn a model from training data. Unsupervised learning methods try to find hidden structures from unlabeled data. Semi-supervised learning models tries to predict patterns in data by making use of both labeled and unlabeled data.

Naive Bayes probabilistic classifiers [78] based on statistical analysis received great interest across research communities and worked exceptionally on document classification tasks. These models learn a probabilistic assumption about how the words in a document are generated. They use a set of training data to estimate the parameters of a model and the entire model is based on the Bayes rule. The Naive Bayes classifier is perhaps the simplest model which assumes that the distribution of different words are independent from each other and aims at finding the posterior probability of a class based on the distribution of words in a document.

Nearest neighbor classifier [30] is a proximity based classifier which uses certain distance measures to perform classification. Decision tree is another supervised learning method which tries to classify documents by subdividing the document based on the presence or absence of a term (word). These subdivisions of a document can be visualized as a hierarchical tree branching based on certain terms in a document. Normally, decision trees have been used in combination with boosting techniques to improve the accuracy of the classifier.

Support Vector Machines (SVM) are supervised learning classification methods make classification based on linear combinations of document features. Word frequency vectors are considered as features of a document and the model tries to learn the coefficients for these features. We can interpret the prediction model as a hyperplane separating different classes. This algorithm tries to find a hyperplane with the maximum distance ϵ (also called as margin) from both positive and negative samples. Since it consists of only linear combinations of a feature vector, it can distinguish among only two classes. However, it can be easily extended for multiple classes using either one vs one or one vs rest approach. Joachims et al [57] has described that text data is an ideal choice for SVM classification due to sparse dimensional nature of text with few irrelevant features.

Ensemble learning combines multiple learning models to improve the accuracy of a problem [62]. It can either contain all learning models of the same method (all decision trees) called as homogeneous ensemble method or learning models of different methods that are stacked on top of one other called as heterogeneous ensemble method. Some strategy like max voting scheme or boosting approaches are used for combining these multiple learners in an ensemble method.

Bagging is based on bootstrap sampling where m samples of data are randomly sampled with replacement from a dataset of size m [15]. Boosting is an iterative process where multiple weak learners are combined to become a strong learner [38, 39]. In this technique, the weak learners later in the iteration tries to pay higher attention to the mistakes made by former base learners. Next, it combines the outputs from all the weak learners in a certain way. Several techniques of combining the results from multiple weak learners exist like AdaBoost [38], XGBoost[23], Gradient Boosting which consider either a different voting rule or trained on different samples of the training data or using gradient descent and updating predictions based on a learning rate.

2.1.2 Deep Learning Techniques

In supervised learning, the training data is associated with a desired response, called labels. A machine learning algorithm classifies the input data into categories by learning discriminant features or attributes based on observed training data. Artificial neural networks (ANN) are one well-known instance of supervised learning algorithms which consists of a multi-layered perceptron (MLP) network. A perceptron (linear threshold units) is a processing unit inspired by biological neurons and multiple perceptrons are connected across layers which try to learn the representations of an input data. These ANNs are the basic building blocks of deep learning which is a modern machine learning technique. Deep learning models have achieved remarkable successes in various applications such as classification [50], object detection [67], natural language processing [27] etc.

The key concept of a deep neural network is learning high-level features from low-level features that can used for various downstream tasks like classification. The first layers of a network represent simple and basic features of the training data and the deeper layers build a complex representation using these low level features. There are multiple deep neural networks available for a wide variety of tasks like Convolution Neural Networks [63] for Image classification, detection, analysis and Recurrent Neural Networks [81] for time series analysis, textual analysis.

A ***MLP*** includes one input layer, one or more linear threshold unit layers (LTU or hidden layers) and one output layer. The information flows from the input layer towards the output layer, hence called feed forward neural networks. Every neuron in a hidden layer receives a weighted input from every neuron within a layer before it. Figure 2.1 demonstrates a MLP with one hidden layer where x_1 and x_2 represent the input features, $W_{1,1}$, $W_{1,2}$ and $W_{2,1}$ represent weights from the weight matrices W_1 and W_2. These matrices associated with the hidden layers(h_1, h_2, h_3) and output layer are used by all the links between two consecutive layers. The vectors b_1 and b_2 represent associated biases. Output of the entire network is represented by Y'. The optimum values for the weight parameters are obtained by an optimization algorithm called backpropagation using gradient descent. The goal is to find the parameters of a network that minimizes the global loss function.

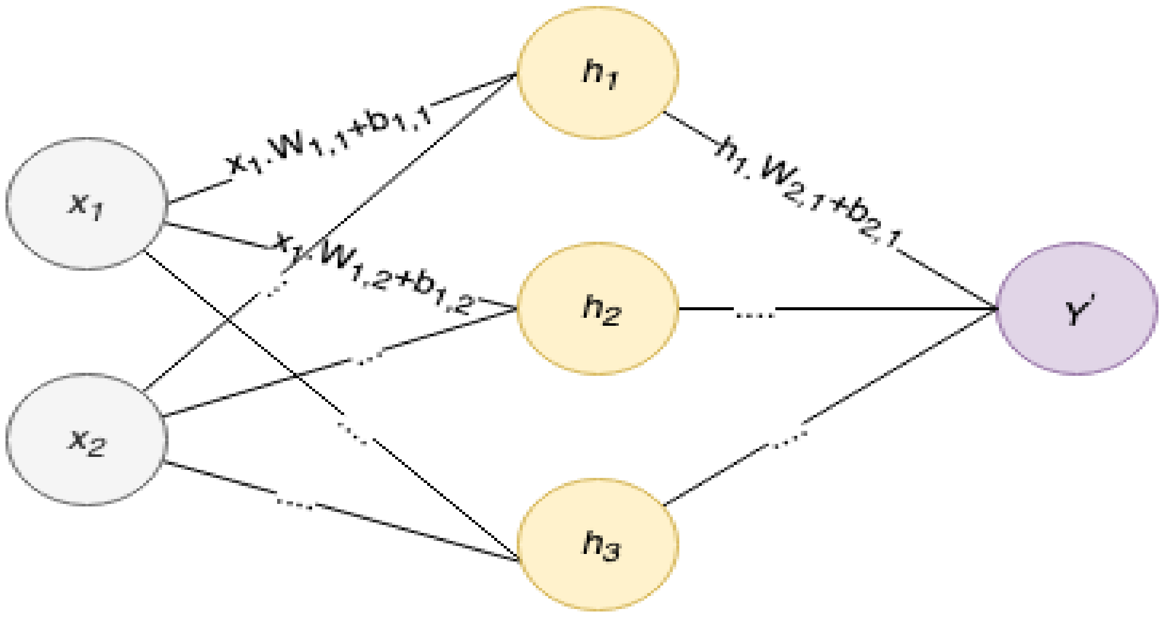


Figure 2.1: A single hidden layer perceptron network.

The ***backpropagation*** algorithm [51] uses gradient descent to find the local minima of the convex error function. It considers the chain rule of calculus to compute the contribution of hidden layers to output and updates the weights associated with hidden layers accordingly. The processing at each neuron can be non-linear and obtained by using different activation functions like rectified linear unit (ReLU)[85] or sigmoid units which are differentiable.

Convolution neural networks (CNNs) [63] are a type of deep neural networks which are widely used in the areas of object recognition, face recognition, handwriting recognition, speech recognition and natural language processing. Generally, CNNs consist of convolution layer, pooling layer and fully connected layer. Convolution layers uses convolutions instead of multiplication to compute the output, resulting in much less inter-connections between layers. The neurons are specialized to respond or activate to stimuli corresponding to a specific location and structure using a set of learnable filters. The output of a neuron is computed by the dot product of the filter and connected local area followed by a non-linear activation. The biggest problem for CNN is that it accepts only fixed length input vector and cannot handle sequences of input of varying length. Figure 2.2 demonstrates the architecture of a Convolution neural network.

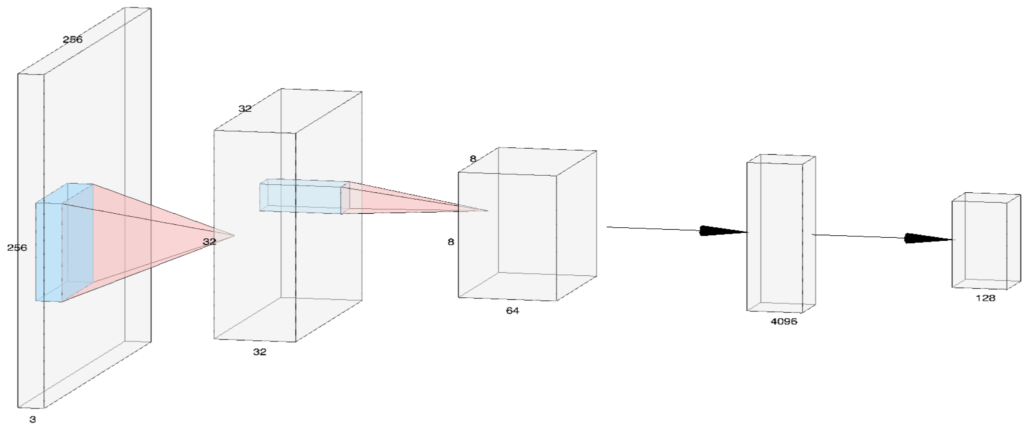


Figure 2.2: A convolutional neural network model

Recurrent neural network (RNN) [81] can not only operate over sequences of input vector, but also generate sequences of output vectors. In contrast to traditional feed forward neural networks, RNN keeps track of its hidden state through recurrent connections which makes it suitable for textual data. In some applications, information in future is also useful and the RNNs can consider future information with a backward pass. Bi-directional RNN considers information from both directions which make a prediction at a certain time step. Figure 2.3 represents a Jordan RNN unit [58] where the hidden states depend on the outputs from previous time steps. The hidden state and output state values are computed using the Equations 2.1 and 2.2.

$$h_t = f_h(W_h.o_{t-1} + W_x.x_t + b_h) \tag{2.1}$$

$$o_t = f_o(W_y.h_t + b_o) \tag{2.2}$$

- x_t : input at time step t
- h_t : hidden state in time step t
- W_h, b_h : Weight and bias parameters corresponding to previous state inputs.
- W_y, b_o : Weight and bias parameters corresponding to output state o_t
- W_x : Weight matrix corresponding to current input x_t
- o_t : output at time step t
- f_h : activation function for hidden state
- f_o : output function

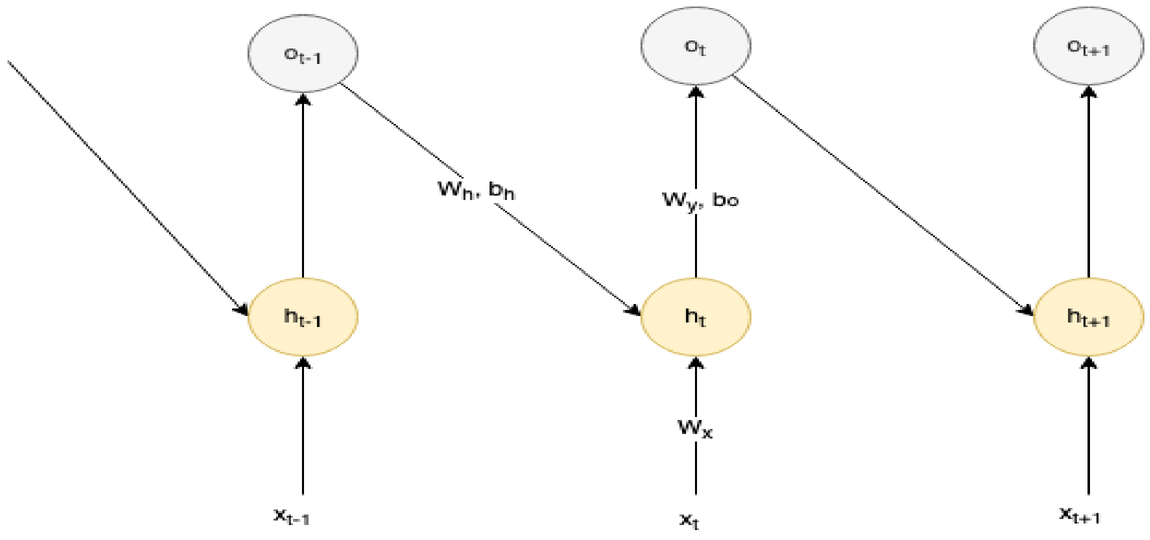


Figure 2.3: Recurrent neural network unit across multiple time steps t

Long Short Term Memory networks (LSTMs)[41] are a type of RNN that can connect previous information to current task. vanilla RNNs suffers from the problem of vanishing/exploding gradient in practice where the information is lost on long sequences. LSTM network better captures the long-term dependencies by using a memory block. This memory block consists of three gates (Input, output and forget) and one cell state. Each gate decides how much information to pass through them using a value between zero and one. The output of each LSTM cell at a time step t is computed using the Equations 2.3 through 2.8.

$$InputGate : i_t = \sigma(W_i.[x_t, h_{t-1}] + b_i) \tag{2.3}$$

$$ForgetGate : f_t = \sigma(W_f.[x_t, h_{t-1}] + b_f) \tag{2.4}$$

$$OutputGate : o_t = \sigma(W_o.[x_t, h_{t-1}] + b_o) \tag{2.5}$$

$$CandidateState : \hat{C}_t = tanh(W_c.[x_t, h_{t-1}] + b_c) \tag{2.6}$$

$$CellState : C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{2.7}$$

$$output : h_t = o_t * tanh(C_t) \tag{2.8}$$

W_i, W_f, W_o are gate coefficients and b_i, b_f, b_o are bias parameters. x_t represents the input at time step t, h_{t-1} represents the hidden state at time step *t-1.* The forget gate f_t uses a weight matrix W_f to forget some of the data from the previous time step t. The output gate combines the information from previous time step with the current cell state C_t. The intermediate candidate state $\hat{C}_t$, which is the information considered from the current time step input using weight matrix W_c and a non-linear function $tanh$.

Generally, in sequence to sequence problems, there is some correspondence between the items in the output and input sequences. However, a generic LSTM network provides equal importance to all items in the input sequence regardless of the output item to be predicted. Attention mechanism [9] tries to solve this issue by providing more importance to certain inputs in the input sequence while predicting an output in the output sequence. This mechanism inherently captures the relations between items in input and output sequences which can be exploited for certain downstream tasks like language translation or textual classification.

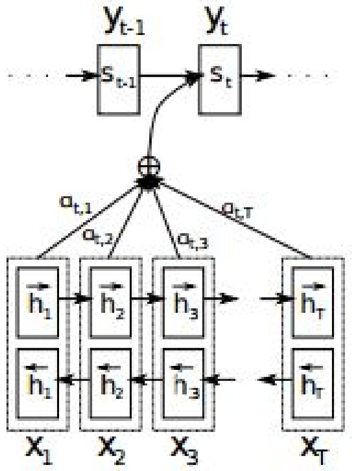


Figure 2.4: Self attention mechanism adapted from Bahadanu et.al.[9]

Self-attention [113] as shown in Figure 2.4 is an attention mechanism which computes a representation of a sequence using different weights to certain portions of the input sequence and has been used successfully in a wide variety of tasks including reading comprehension, textual entailment and task-independent sentence representations. The goal of self attention in natural language processing algorithms is to learn the dependencies between words in a sentence and use that information to capture the internal structure of a sentence. With a self attention mechanism, the network can attend to a state produced many time steps earlier which means that the latest state does not need to store all the information.

These deep learning and machine learning algorithms are utilized for analyzing text. In particular, these algorithms are used for classifying text documents using a sequential model like RNN or using context level information in text using attention mechanisms. Sections 3.2.2 and 3.2.3 describe in detail about the usage of these algorithms for text classification. Also, many of the existing natural language processing (NLP) techniques use these models and algorithms for classifying words in to entities as described in Section 2.2.2, or for obtaining dependency relations among words in a sentence as described in Section 2.2.3.

2.2 Natural Language Processing Techniques

Natural language processing (NLP) is the study of a variety of topics ranging from computational processing to understanding human languages. These techniques are widely used in many domains such as health care document analysis [56], event extraction [117], language translation [9], and Image understanding [123]. Previous techniques focused on the usage of human-crafted rules and certain linguistic features but with the advent of data driven computing, current techniques use statistical models, and advanced artificial neural networks for language understanding [89]. The deep neural network models have significantly performed better than previous models across various tasks like named entity recognition [84], text summarization [10], dependency parsing [22].

2.2.1 Word Embeddings

The representation of words and documents as dense, distributed, fixed-length word vectors built using word co-occurrence statistics as per the distributional hypothesis [48] that have an appealing and intuitive interpretation are called word embeddings [5]. These representations encodes the syntactic and semantic relationships among words and are used in many machine learning algorithms and strategies. The vector space model (VSM) [101] was the most influential model for encoding words and documents, but recent models try to capture the semantics of a word based on its usage using advanced neural networks.

Word Embeddings are generally categorized into two types, methods which use *local* data (context) and methods that use *global* data (words counts and frequencies) [16]. Embedding models derived from neural network models leverage language models, which predicts the next word given its context also called as prediction based models [82]. Statistical techniques model the probabilistic distribution of words [8] in a language using co-occurrence matrices or frequency counts of words also called as count based models.

Word2Vec [82] is a simple and efficient architecture that provides interesting semantic properties. Continuous bag of words (CBOW) and skipgram are the two widely used Word2Vec models. The CBOW architecture consists of a feed forward neural network model that aims at predicting the current word based on its surrounding context as shown in Figure 2.5. The architecture consists of an input layer of size of the word vocabulary, which encodes a target word based on the combinations of one hot vector representations of surrounding words. The output layer contains a one-hot vector of target word and has same size as input layer.

The skip-gram model tries to predict the surrounding context words given a target word and is similar to CBOW model in training and representation vector sizes. Figure 2.6 represents the visualization of words reduced to two dimensions using t-SNE. These representations can capture the semantics of a word and are widely used in a variety of downstream NLP tasks.

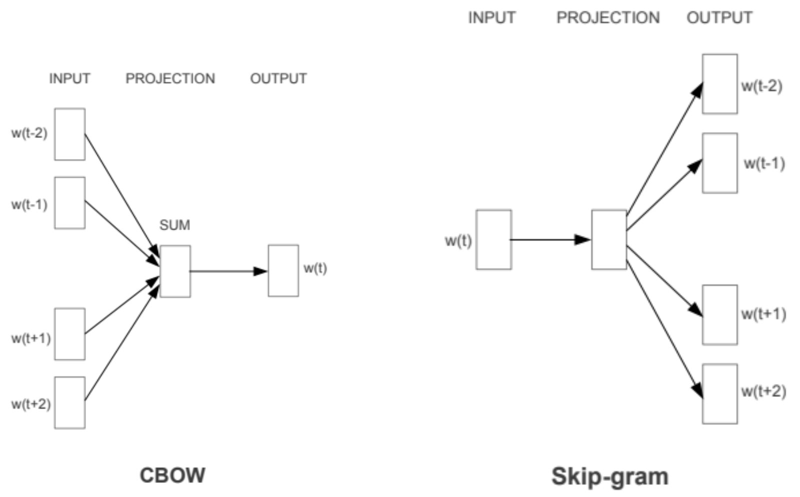


Figure 2.5: A word2vec model adapted from Mikolov et.al. [82]

GloVe [92] is another prominent word embedding architecture which combines the best of both prediction based and count based models. It combines global matrix factorization and local context window methods through a bi-linear regression model. Recent models exploit various lingustic features like dependency parse-trees, symmetric patterns, leveraging subword patterns, representing words as probability distributions, or character based embeddings [3, 19, 29].

Recent NLP works, include the usage of pre-trained sentence level embeddings for strong prediction models on various downstream tasks like classification using transfer learning. The universal sentence encoder [19] uses transformer networks introduced in [113] to obtain higher accuracy for semantic textual similarities task. Infersent [29] is another sentence level embedding network that is trained using natural language inference data using self-attentive bi-directional LSTM network.

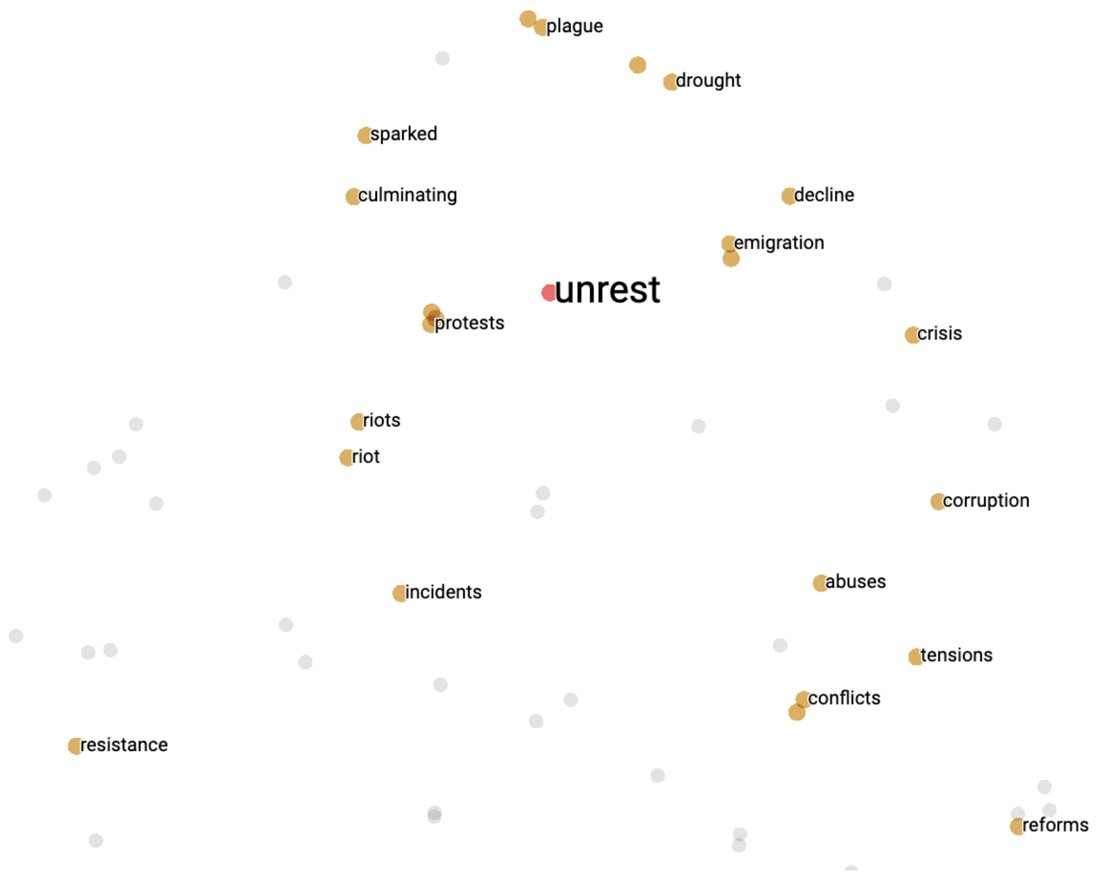


Figure 2.6: Visualization of word embeddings in 2 dimensions using T-SNE

2.2.2 Named Entity Recognition

Named entity recognition (NER) is a sub-task of information extraction that identifies named entities and classify them into categories such as person, location, organization, time etc. NER systems are often used for question answering, information retreival, co-reference resolution etc. Early NER systems focused on the usage of linguistic rules, lexicons and ontologies [124]. They were followed by NER systems developed using feature engineering and supervised machine learning models [84].

Hidden markov models (HMM) [95], support vector machines (SVM) [57], conditional random fields (CRF) [65] are the common machine learning systems for NER which are trained on example inputs along with expected outputs. Malouf et.al. [75] considered HMM with multiple features like capitalization, whether a word was first in a sentence, a collection of first and last names from various dictionaries. Carreras et.al. [18] used bag of words, trigger words, binary relations among previously predicted labels as additional features. A SVM model with multiple window sizes, features (orthographic, prefixes, suffixes) from neighboring words, weighing neighboring word features based on their position for prediction was proposed by Li et.al. [69].

Stanford NER [36] uses a linear chain of CRF sequence models based on viterbi algorithm [37] for predicting labels to entities. Recent models are based on recurrent neural networks (Bi-LSTMs) using word embeddings and POS tags for NER prediction [53, 102]. Character level architectures predict labels for each character using CNN models which are then transformed to word labels using another LSTM network or CRF [64, 70] . Current state-of-the art models combine both character level and word context for predicting NER labels [34]. Yadav et.al. [125] implemeted a model that augments character+word NN architecture with affix features.

Akbik et.al. [3] proposes contextual string embeddings for annotating named entities. The model represented in Figure 2.7 takes a sequence of characters as input to a bidirectional character level neural language model which provides contextual string embeddings. These embeddings are then used in a Bi-LSTM CRF sequence labeling model to obtain named entities.

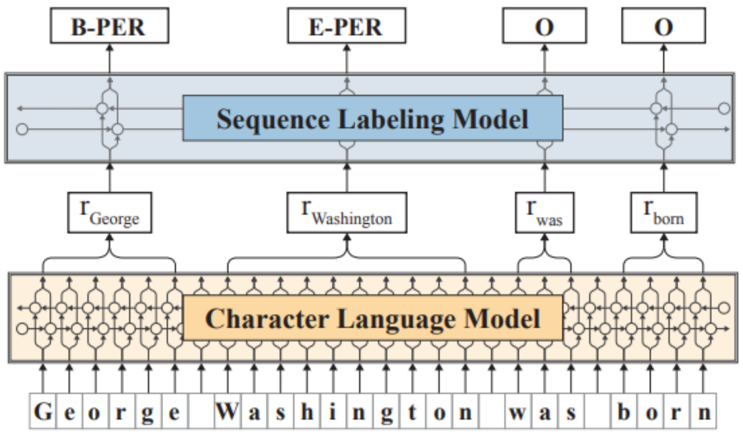


Figure 2.7: Model overview for NER adapted from Akbik et.al. [3]

2.2.3 Dependency Relations

Dependency structure of sentences describe the relationship between words as a tree structure depicted as arrows going from the head to the dependent as shown in Figure 2.8. Dependency parsing is the task of analyzing the syntactic dependency structure of an input sentence. Transition-based parsing and neural dependency parsing are two widely used models for obtaining a dependency tree [76]. A transition-based parsing tries to learn a model which can predict the next transition in the state machine based on the transition history such that these sequence of transitions can construct a dependency tree for an input sentence. A greedy deterministic transition based parsing mechanism using shift and arc-standard operations was proposed by Nivre et.al [86].

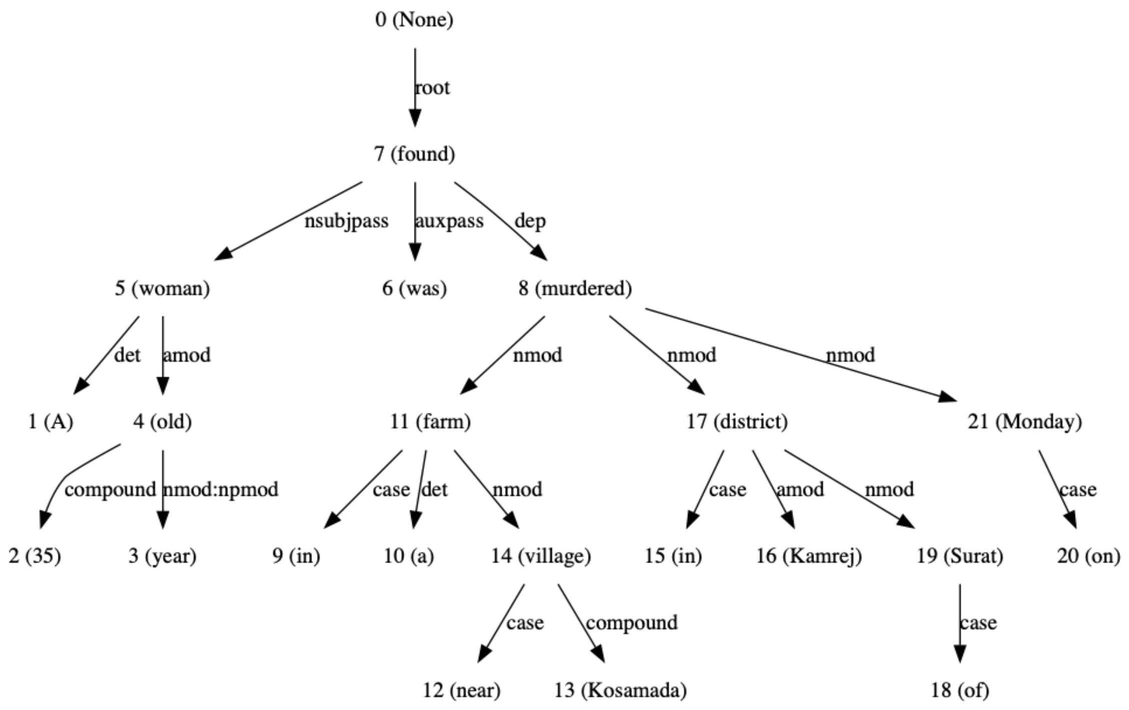


Figure 2.8: Dependency tree for the sentence: *A 35 year old woman was found murdered in a farm near Kosamada village in Kamrej district of Surat on Monday.*

Greedy, transition-based neural dependency parsers perform significantly better compared to other traditional feature based discriminative dependency parsers [76]. Model proposed by Chen et.al. [22] uses various input features like vector representations of words, parts-of-speech (POS) tags for those words and the arc labels for some words in a sentence S. Graph-based dependency parsers compute a score for every possible dependency from every word to every other word [76].

2.2.4 Coreference Resolution

Coreference resolution is the task of finding all expressions that refer to same entity in a text and grouping them, which is essentially a clustering problem [74]. It is useful for a lot of higher level NLP tasks such as document summarization, question answering and information extraction. Mention detection is the first step of coreference resolution where spans of text (pronouns, Named entities, Noun phrases) referring to an entity are identified. The second step is to link these entities using either rule-based methods or other clustering-based mention ranking methods.

Many coreference systems operate by linking pairs of entity mentions together using a binary classifier that predicts whether an entity is co-referent to every other entity. Current research [26, 35, 120] use a neural network model to build distributed representation of pairs of coreference clusters. It follows an incremental approach where each entity is considered as its own cluster at the start and the network decides to merge different clusters using cluster ranking procedures [96, 105].

2.3 5Ws Extraction Techniques

Analyzing text to extract information corresponding to an event is a fundamental task for various use cases. News summarization [88], event linking [111] and news aggregation are used by researchers from social sciences to better understand an event. Main event extraction from news is a fundamental task in news analysis [90] where some techniques use explicit descriptors by employing topic modeling and clustering [108]. Other approaches include using polarity, sentiment [40] to capture an end user's perspective. Other approaches use event linking [32] to group shared events or focus on specific information like crisis monitoring [108].

Event extraction is used to automatically identify events in text and to derive detailed information such as time, location, participants and their roles in the events [33]. Journalistic 5Ws extraction tries to answer the 5W questions: *where, when, who, what* and *why* corresponding to an event and are closely related to domain question answering systems. The 5Ws task is closely related to semantic role labeling (SRL) [17] which identifies arguments in an article and tries to assign roles to them [91].

Most approaches follow the model proposed by Gildea et.al. [43] which uses heuristics to resolve inconsistencies. Multiple models [79, 106] are available for SRL in various languages which are built using supervised learning techniques. Message understanding conference (MUC) consists of research related to information extraction from events using knowledge engineering, statistical machine learning and natural language processing.

Previous work on 5Ws extraction included verb-driven approach [32, 91] to extract event semantic information. Assigning semantic role labels to nouns was proposed by Wang et.al. [116] for chinese text. Other approaches use lexico-syntactic features like POS and morphological features for identication of 5Ws [31]. Hierarchical clustering approach to groups news articles based on content similarity and use two-stage SRL [71] was proposed by Parton et.al [91]. A visualization-based system [31] that tracks sentiment with textual summary and polarity was also used to better understand an event. 5WTAG [128] is an algorithm for detecting microblog topics based on model of 5Ws. Sharma et.al. [103] developed a fine grained approach for extracting 5Ws using named entities, co-reference resolution and supervised classifiers. The concept of 5Ws introduced in [33] are described below.

- $Where$? The location of an action/event.
- $When$? The temporal component about an event.
- $Who/Whom$? Animate/inanimate agents involved in an event.
- $What$? The resulting action of an event.
- Why? The semantic concept for the occurrence of an event.

A combination of supervised learning approaches are also used for event extraction by dividing the extraction task into a series of supervised machine learning subtasks to evaluate importance and difficulty of each task [2]. Yaman et.al. [126] uses three independent subsystems to extract 5Ws answers which are then combined using a trained SVM model. Even though many models [60, 126] utilize supervised learning models for 5Ws question answering, it is difficult to obtain ground truth data for domain-specific articles. In this thesis, we focus on lexico-syntactic features in combination with domain knowledge to better extract the 5Ws corresponding to an event from text.

Where extraction has been widely researched as discussed below. Forecasting civil unrest by extracting information from textual data including news articles, tweets and facebook has been studied in [122] and [28]. Embers [83, 97] is a forecasting system developed using various data sources and human analysts for preparing a Gold Standard Report (GSR). This system uses a probabilistic approach for predicting future events by extracting location and time information from text. Geocoding is done using the location information provided by either tweets or facebook posts. For news articles a probabilistic model that uses a gazetteer and a set of rules is used. The information extracted from an article is then cross-references with all similar articles using a clustering approach. This thesis focuses on extracting location information rather than geocoding it.

Extracting ***why*** is hard using only domain knowledge and is based on implicit causal relationships [7]. Causality has been extensively studied in multiple disciplines, including psychology [121], linguistics [107], philosophy [119] and computer science [6] classified causal links into four types: adverbial links (so, hence), prepositional links (because of, on account of), subordination (because, as, since) and clause integrated links (that's why, the result was). Causation adverbs, causal links, causative verbs are used as indicators for causal relationships in text [61]. Some approaches use classification of syntactic representations of causal relations [13, 44] wheras some other approaches use semantic analysis[100]. Automatically discovering lingustic patterns expressing causal relations by focusing on explicit syntactic patterns of the form *NP1-CausativeVerb-NP2* is studied in Girju et.al [44]. $NP1$ and $NP2$ are the noun phrases which are connected by a causative verb. Further information about causal relationship extraction can be found in [7].

Chapter 3

Methodology

Current methods to extract 5Ws typically involve manual analysis of text to obtain related information or automated approaches which are domain independent. In this thesis, we developed an approach that obtains the Journalistic 5Ws customized for the specific domain of social unrest. It should be noted that much of our proposed approach is domain independent. Therefore, the approach can be extended to any domain that can be represented by a vocabulary, e.g., human rights violation.

Section 3.1 describes an overview of the entire pipeline for extracting 5Ws. Section 3.2 includes details about the classifiers used for classifying social unrest related articles and Section 3.3 outlines the approach for extracting each individual W in detail.

3.1 A Framework to Extract Journalistic 5Ws

Our framework is based on a set of heuristics derived from the structure of the articles, linguistic features of the sentences in the article and journalistic principles. These principles are briefly described below. Our goal is to make our approach automated with little or no user interaction.

- *Domain specific cues:* Uses a list of domain-specific keywords to better capture the importance of a W candidate to the domain. If a candidate is surrounded by more domain-related words, then it is more likely to better represent the W.

- *Document structure:* Generally, sentences at the top of a document contain more important, related information. This approach leverages this principle as a heuristic while ranking candidates.

- *Linguistic cues:* Semantic similarity and the dependency structure of a sentence provide linguistic features that are utilized for identifying and ranking candidates.

- *Journalistic principles:* The important information is described at the top of a document by journalists. Certain principles used by journalists for reporting an event are incorporated while ranking candidates.

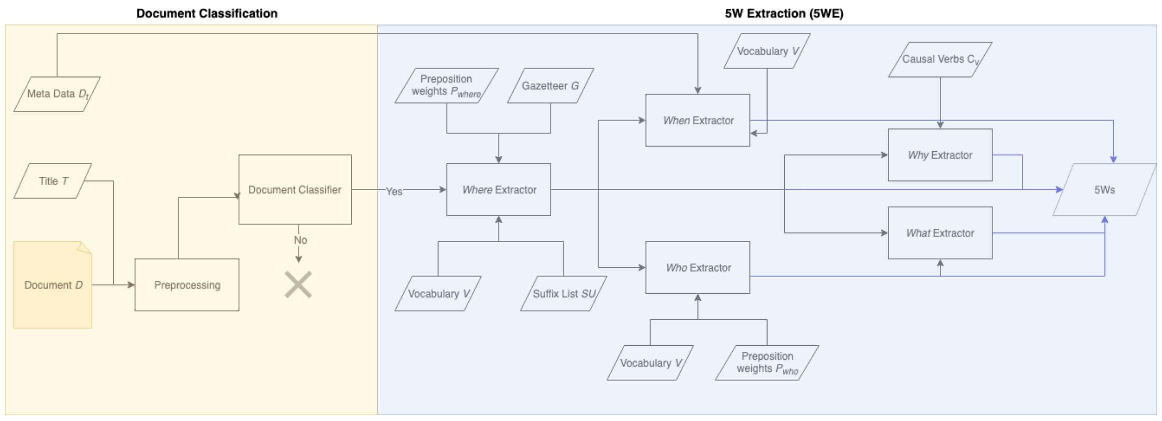


Figure 3.1: Overview of the 5Ws extraction framework

A schematic of our framework for Journalistic 5Ws extraction is shown in Figure 3.1. The framework is divided into two major steps.

- *Document classification:* Decides whether the text is related to a specific topic or not using a pre-trained classifier. If it doesn't belong to a specific topic, it isn't analyzed any further.

- *5Ws extraction:* Consists of a combination of multiple NLP techniques which outputs the Journalistic 5Ws corresponding to the input text which is further divided in 5 different sub-steps for each W, which interact with each other.

Algorithm 1 provides the pseudo code for our overall approach. In the remainder of this thesis, we use document and news articles interchangeably.

Algorithm 1 5Ws Extractor (5WE)

```
1:  $D$: Document
2:  $V$: List of Domain-specific words
3:  $G$: Gazetteer (List of Locations)
4:  $P_{where}$: Preposition Scores for *where*
5:  $P_{who}$: Preposition Scores for *who*
6:  $D_t$: Published date
7:  $C_v$: Causal verbs and adverbs
8:  $T$: Title
9:  $Tp$: Topic/Domain
10: $R$: Domain specific neighborhood radius
11: $SU$: List of place suffixes
12: procedure 5Ws Extractor($D, V, G, P_{where}, P_{who}, D_t, C_v, T, Tp, R, SU$)
13:     $D' \leftarrow preprocess(D)$
14:     $Class \leftarrow documentClassifier(D')$
15:     if $Class = Tp$ then
16:         $where \leftarrow whereExtractor(D, V, G, P_{where}, T, R, SU)$
17:         $D_w \leftarrow best(where)$
18:         $who \leftarrow whoandwhatExtractor(D, D_w, T, P_{who}, V, R)$
19:         $when \leftarrow whenExtractor(D, D_t, D_w)$
20:         $why \leftarrow whyExtractor(D, C_v, D_w)$
21:     end if
22:     $5Ws \leftarrow \{where, when, who, what, why\}$
23:     return $5Ws$
24: end procedure
```

3.2 Development of a Social Unrest Document Classifier

Document classification is the process of classifying a document based on its contents into two or more classes. A typical document classification process involves collection of documents from a multitude of data sources, pre-processing the documents, feature selection and representation, classification and evaluation. Next, we describe a collection of statistical and deep neural networks that are used for classifying textual documents.

Preprocessing involves representing text in clear word format which is obtained by removing insignificant characters, commonly used stop words (a, an, the, and), converting a word to its canonical form using a stemming algorithm, and converting the data to unicode string format. Tokenization and Lemmatization are the two major algorithms used in the preprocessing step. In the next subsections, we describe about the ways of representing documents and two of the classification approaches that are used for evaluation of social unrest classifier because of their potential improvements from traditional approaches described in Chapter 2.

3.2.1 Document Representations

Textual data in a document must be represented as numerical data for the purpose of classification using traditional machine learning algorithms. Word representations as described in Chapter 2 can be used for representing documents which also capture the semantic context of words in a document. Different ways of representing an entire document are described below:

- *Term frequency - Inverse document frequencies (TF-IDF)* are used for representing a document in a corpus.

- *Word2Vec* [81] and *GloVe* [92] embeddings of words can be combined by either averaging or taking the maximum of all word representations in a document.
- *Doc2Vec* [66] is used for representing an entire document as fixed length vector.

The next step involves using a classification algorithm on the pre-processed vector space data. The simplest classification algorithm would be to classify documents based on domain-specific vocabulary. If a document contains at least k domain-specific words, then it is classified as belonging to that domain. The drawback of this approach is that it doesn't consider the context of a document and fails to interpret homonyms. Other techniques like latent dirichlet allocation (LDA) and latent semantic analysis (LSA) include using word document frequencies in the form of a matrix and using singular vector decomposition (SVD) to classify documents. [94, 110, 118]

Modern classification algorithms use machine learning techniques to combine different words in a document and make a prediction based on the combined features. The inputs provided to a machine learning model can be either TF-IDF vectors or word embeddings (Word2Vec, GloVe, Doc2Vec) which are then combined to produce a final representation of the entire document used for classification. Different ensemble techniques like majority vote bagging, boosting and stacked classifiers can also be used for increasing the performance of a classifier. Some classifiers even consider the sentiment (vader [54], textblob[73]) as an additional feature while classifying documents.

3.2.2 Fasttext Classification

Many continuous representation of words models represent words using a distinct vector, without parameter sharing and ignore the internal structure of words. [14]

proposed to learn representations from character n-grams which considers subword information. Joulin et.al. [59] proposed a hierarchical model for text classification named ***Fasttext*** with a fast loss approximation. Their model consists of a classifier with shared parameters so as to generalize in the context of large output space. A softmax function is used to compute the probability distribution, for minimizing the negative log likelihood over classes as shown in Equation 3.1.

$$\frac{-1}{N}\sum_{n=1}^{N} y_n log(f(BAx_n)) \tag{3.1}$$

where x_n is the normalized bag of features, N is the number of documents, y_n is the label, A and B are the weight matrices of size $n \times n$ and f is the softmax function to compute the probability distribution over predefined classes. This model uses a hierarchical softmax function for reducing the training and inference time. Each node in the network is associated with a probability that is the probability of the path from root to that node. If a node is at depth $l+1$ with parents $n_1, ..., n_l$, its probability is given by Equation 3.2.

$$P(n_{l+1}) = \prod_{i=1}^{l} P(n_i) \tag{3.2}$$

3.2.3 Hierarchical Attention Network

An effective neural network based approach for document classification based on attention mechanism was proposed in [127]. Their model takes advantage of the inherent structure of a document, where a set of words form a sentence and a group of sentence form a document. It also considers the importance of certain words and sentences depending on the context for classification using an attention mechanism [113] at both word level and sentence level. The architecture of Hierarchical attention network (HAN) is shown in Figure 3.2 which consists of a word level attention layer, a sentence encoder, a sentence level attention layer and a softmax unit at the end for classification. Gated recurrent units (GRU) [25] are used to track the state of sequences and consists of two gates: reset gate r_t and update gate z_t which together control how information is updated at each time step h_t.

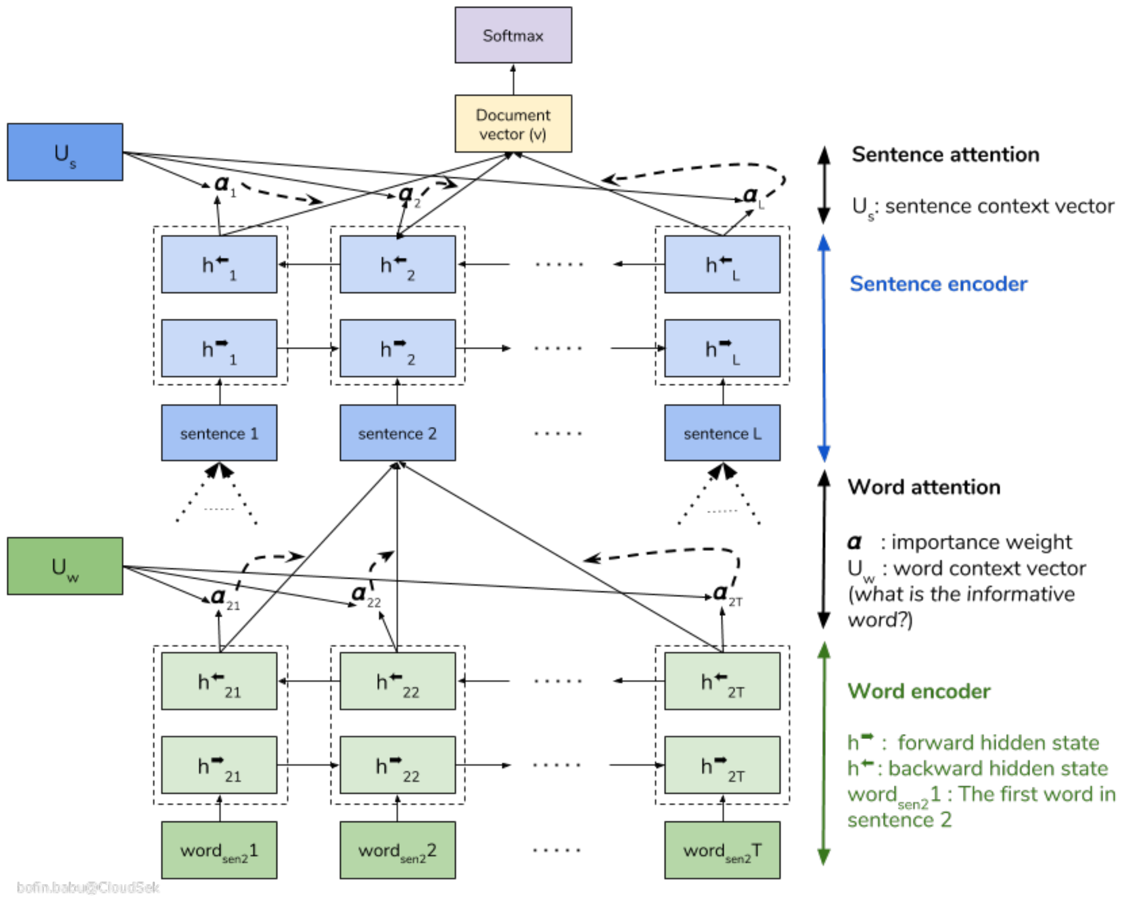


Figure 3.2: Model architecture of hierarchical attention network

Since not all words contribute equally to a sentence meaning, attention mechanism to extract important words is used and the aggregation of these informative words form a sentence vector. First a word representation h_{it} is fed to a one layer MLP to get u_{it}. Then the importance of this word u_{it} with respect to the word level context vector u_w is calculated to obtain a normalized weight α_{it} through a softmax function. Finally, the sentence level representation is obtained by the weighted sum of all word representations. All the representations u_w, u_{it} are jointly learned during the training process. Sentence level attention is also done using the similar process with a sentence level context vector u_s. The final document vector v is used for classification based on a softmax function and trained using negative log likelihood. The attention mechanism used in this model is given by the Equations 3.3 through 3.5.

$$u_{it} = tanh(W_w h_{it} + b_w) \tag{3.3}$$

$$\alpha_{it} = \frac{exp(u_{it}^{\top} u_w)}{\sum_t exp(u_{it}^{\top} u_w)} \tag{3.4}$$

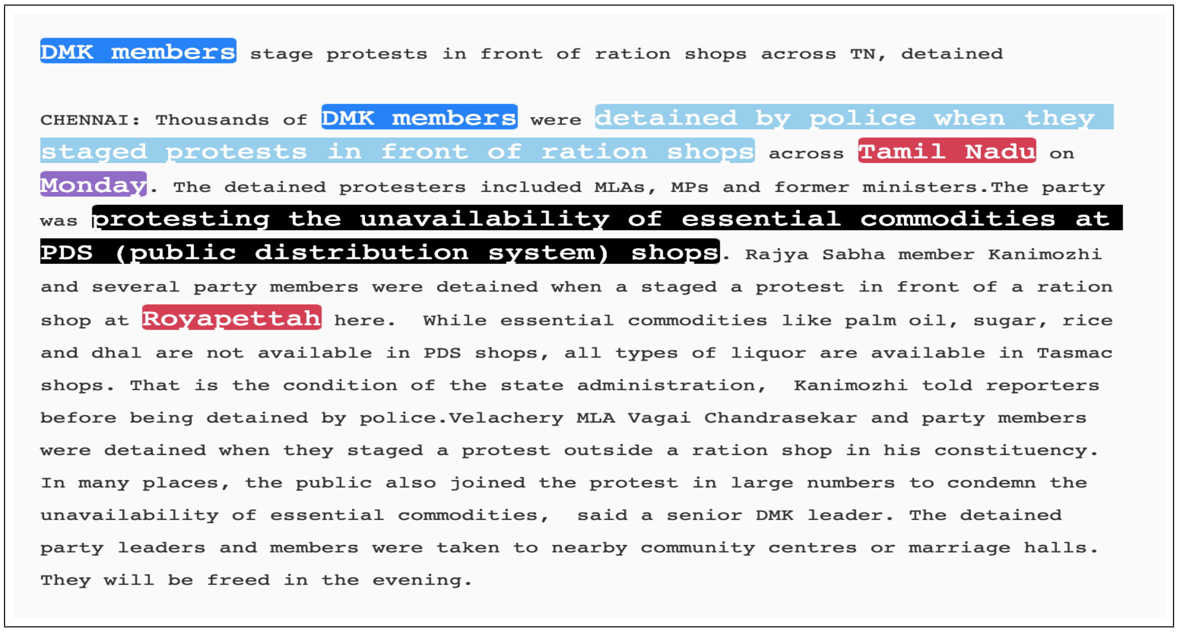
$$s_i = \sum_t \alpha_{it} h_{it} \tag{3.5}$$

The approaches discussed in the Section 3.2.2 and 3.2.3 are used for document classification tasks to provide better accuracies. The Fasttext model described has fast inference time with high accuracy values for most text classification tasks. The HAN model considers the inherent document structure in the classification process. These models are developed and tested using an unrest related documents dataset in Section 4.2.1.

3.3 5Ws Extraction (5WE) Approach

The extraction of Journalistic 5Ws is performed one W at a time. The input to this step is a text document classified as relevant to a specific topic/domain. The entire process begins with co-reference resolution where all the pronouns in an article are replaced with their corresponding noun phrases as described in Section 2.2.4. This step is important as it helps the model in better understanding the importance of a word in the entire article. The output of this step are the phrases that best represents the answers to Journalistic 5Ws as represented in Figure 3.3. The other inputs that the extraction process considers are specific to individual Ws as described in Algorithm 1.

- Meta data: Date of publication D_t of the article.
- Gazetteer G: A list of locations corresponding to a geographical region.
- Preposition scores P_{where}: Weights for prepositions located before locations.
- Preposition scores P_{who}: Weights for prepositions located before person names.
- Causal verbs, conjunctions C_v: A curated list of verbs and conjunctions associated with causation.
- Domain-specific vocabulary V: A list of keywords related to the domain.



DMK members stage protests in front of ration shops across TN, detained

CHENNAI: Thousands of DMK members were detained by police when they staged protests in front of ration shops across Tamil Nadu on Monday. The detained protesters included MLAs, MPs and former ministers.The party was protesting the unavailability of essential commodities at PDS (public distribution system) shops. Rajya Sabha member Kanimozhi and several party members were detained when a staged a protest in front of a ration shop at Royapettah here. While essential commodities like palm oil, sugar, rice and dhal are not available in PDS shops, all types of liquor are available in Tasmac shops. That is the condition of the state administration, Kanimozhi told reporters before being detained by police.Velachery MLA Vagai Chandrasekar and party members were detained when they staged a protest outside a ration shop in his constituency. In many places, the public also joined the protest in large numbers to condemn the unavailability of essential commodities, said a senior DMK leader. The detained party leaders and members were taken to nearby community centres or marriage halls. They will be freed in the evening.

Figure 3.3: A sample article representing the annotation of 5Ws. Blue color represents the *who*, purple color represents the *when*, red color represents the *where*, light blue color represents *what* and black color represents *why*. [87]

Extraction of individual Ws consists of two tasks:

- *Candidate identification* task is the extraction of potential candidates for a Journalistic W from text. This task makes use of several existing techniques like Named Entity Recognition, Parts-of-Speech tagging, Dependency Parsing and Co-reference Resolution for identifying candidates.
- *Candidate ranking* is ranking these potential candidates and identifying the best candidate that closely represents the answer to a specific Journalistic W using semantic cues, other topic/domain-specific pointers in the text document. Some sub-steps make use of the results from the previous sub-steps to obtain accurate results as shown in Figure 3.1.

Using prepositions for providing additional weights to *where* and *who* candidates is useful as it provides additional confidence about a candidate being a W based on the syntactic structure of a sentence. The presence of a word after a certain preposition in a sentence provides a sense about the meaning of the word and its context. For example, in most cases *near* is a preposition most likely to be associated with a location. Our approach leverages this intuition about prepositions for scoring candidates. The preposition scores are collected using a large customized set of articles related to the domain that are being examined.

These scores are customized as different sources might use different prepositions and some prepositions have become obsolete with time. For example, *near* is most likely to be used by New York Times compared to The Hindu. These insights to preposition likelihood scores should be obtained by using a customized set of articles related to the domain and geographical scope. If a word is tagged as the required entity by a Named Entity Recognizer then a preposition preceding the word is analyzed to get a score as described. This score calculates the conditional probability that a word is a specific entity w_e when preceded by a preposition p. The score is defined as: How often is the word following a particular preposition p a specific entity e is represented by P as shown in Equation 3.6.

$$P = \frac{N_{p,w_e}}{N_p} \tag{3.6}$$

where N_{p,w_e} is the number of times a preposition p occurs before a word w_e of entity type e. N_p is the number of times preposition p occurs in the entire set of articles.

The results from this process can be improved by using a better co-reference system, including additional weighted domain-specific words, using techniques of the question answering systems [114].The remaining subsections describe in detail the candidate identification and ranking for individual Ws.

3.3.1 Extraction of Where

In this step, we derive an ordered set of locations that best represents the answer to the first Journalistic W, i.e., *where* in an article. The inputs to this step are the text document D, a list of topic/domain-specific words V, a list of locations G and a list of prepositions P_{where} along with their likelihoods that are closely associated with a location. It produces an ordered set of location candidates and their scores. However, this sub-step doesn't correspond every ranked location to a specific event if multiple events are mentioned in a single article.

The candidate identification task identifies potential candidates for *where*. We first perform a customized Named Entity Recognition on the entire article and obtain words that are tagged as either *Location* or *Organization*. We then consider words that contain a suffix from a specific set of suffixes as candidates. This set of suffixes is

Algorithm 2 *Where* Candidate Identification

```
1: $D$: Document
2: $SU$: List of place suffixes
3: procedure CANDIDATE IDENTIFIER($D, SU$)
4:     $D' \leftarrow$ preprocess(D).
5:     $C \leftarrow \{\}$
6:     $Words : W \leftarrow wordTokenizer(D')$
7:     $Sentences : S \leftarrow sentTokenizer(D')$
8:     $N \leftarrow length(W)$
9:     for each $w_i \in W$ do
10:        if $NER(w_i)$ is $LOC$ or $ORG$ then
11:            $C \cup \{\{w_i, i, S(w_i)\}\}$
12:        end if
13:        if suffix($w_i$) in $SU$ then
14:            $C \cup \{\{w_i, i, S(w_i)\}\}$
15:        end if
16:    end for
17:    return $C$
18: end procedure
```

prepared by analyzing the location names from a specific geographical area of focus. We then merge adjacent candidates to obtain candidate phrases and add them to the set of candidates for *where*. The output of the first task will be a set of candidates for *where* along with some additional information like the word number of a candidate, the sentence number in which a candidate is present, and the source from which it is obtained (Named entity recognition or Suffix list).

The candidate ranking task ranks these potential candidates. We consider several factors for scoring a candidate and assume that the most important candidate will be mentioned early and often in the text document. We also assume that the most important candidate will be surrounded by some specific set of words from a vocabulary that are specific to a topic/domain. We consider six different factors for scoring a candidate as described below.

Algorithm 3 Extraction of $Where$

```
 1: $D$: Document
 2: $V$: Vocabulary (List of Domain-specific words)
 3: $G$: Gazetteer (List of Locations)
 4: $SU$: List of place suffixes
 5: $P_{where}$: Preposition likelihood scores for $where$
 6: $T$: Title of the document
 7: $R$: Neighborhood radius
 8: procedure $where$EXTRACTOR($D, V, G, P_{where}, T, SU, R$)
 9:     $D' \leftarrow$ preprocess(D).
10:     $Words : W \leftarrow wordTokenizer(D')$
11:     $Sentences : S \leftarrow sentTokenizer(D')$
12:     $N \leftarrow length(W)$
13:     $C \leftarrow candidateIdentifier(D, SU)$                ▷ Algorithm 2
14:     $Finalscores : FS \leftarrow \{\}$
15:     for each candidate $c_i \in C$ do
16:         $s_w \leftarrow 1 - \frac{i}{N}$
17:         $s_s \leftarrow cosinesimilarity(S_i, T)$
18:         if $c_i \in T$ then
19:             $s_t \leftarrow 1$
20:         end if
21:         if $c_i \in L$ then
22:             $s_g \leftarrow 1$
23:         end if
24:         if $W[i-1] \in P_{where}$ then
25:             $s_p \leftarrow P_{where}[W[i-1]]$
26:         end if
27:         $Vwords \leftarrow \{\}$
28:         for $r$ in $range(1, R)$ do
29:             if $W[i+r]$ or $W[i-r]$ in $V$ then
30:                 $Vwords+ = \{W[i+r], r\}$
31:             end if
32:         end for
33:         $s_v \leftarrow Sum(Vwords)$                ▷ weighted sum function
34:         $FS+ = \{c_i, Locationscore(s_p, s_g, s_t, s_w, s_s, s_v)\}$
35:     end for
36:     return $FS$
37: end procedure
```

Word position score s_w is used for ordering candidates c_i based on their position in the entire document. Based on inverse pyramid concept [24], important things tend to be mentioned at the top of an article and hence must receive higher score compared to candidates later in the article. This principle is captured by this score and shown in Line 16 of Algorithm 3.

$$s_w(c_i) = 1 - \frac{i}{N} \tag{3.7}$$

where i represents the position of a candidate and N represents the total number of words in a text document.

Sentence similarity score s_s captures the importance of the sentence s_{c_i} in which a candidate c_i is present with respect to the entire document. As title T, better represents the overall summary of a text document, we use the semantic similarity between a sentence s_{c_i} and the title T. We use cosine similarity to compute the similarity between different embeddings. These embeddings can be obtained using any context level emebedding techniques such Universal sentence encoder (USE) [19] which is pre-trained on a large dataset collected from various sources like wikipedia, discussion forms and web news. This score is shown in Line 17 of Algorithm 3.

$$s_s(c_i) = similarity(USE(s_{c_i}), USE(T)) \tag{3.8}$$

Title score s_t captures the existence of a candidate in the title of a document T. The underlying principle is that important actors, locations related to an event are usually mentioned in the title of a document. If the candidate c_i is present in the title, then a Boolean score of 1 is assigned, 0 otherwise. The title score s_t for a candidate

c_i is calculated as shown in Lines 18-20 of Algorithm 3 :

$$s_t(c_i) = \begin{cases} 1 & c_i \in T \\ \\ 0 & c_i \notin T \end{cases} \tag{3.9}$$

Gazette score s_g captures the existence of a candidate in a list of location G. If a candidate c_i which is not actually a recognized location is extracted as candidate, then a lower score is assigned to it based on this gazetteer G. Since, location names might be misspelled or modified over time, we use a fuzzy string matching approach[45] for searching a candidate in the gazetteer. This fuzzy string matching tries to capture location names which differ by only 1 or 2 characters and generalizes to almost all locations. For example, *Bangalor* and *Bengalur* are both considered a legitimate candidates and they both differ in just 2-3 characters to the actual location *Bangalore*. The list of location G can be prepared by combining available national gazetteers or other sources like Google maps. We use a customized set of locations obtained from the census of India and National Geo-spatial Intelligence Agency (NGIA). The gazette score s_g is computed as shown in Lines 21-23 of Algorithm 3:

$$s_g(c_i) = \begin{cases} 1 & c_i \in G \\ \\ 0 & c_i \notin G \end{cases} \tag{3.10}$$

Where preposition scores s_p are computed using a list of prepositions P_{where} computed using a data driven approach. This score provides additional confidence to candidates c_i based on the preposition preceding it. This score helps in avoiding candidates which aren't exactly locations but are recognized as candidates in the candidate selection process. If a candidate c_i is preceded by a preposition p_j from a list of prepositions P_{where}, then the score corresponding to p_j is provided to c_i. Some of the common prepositions associated with locations are *near*, *at* as presented in Section 4.2.2. The process for computing preposition scores is described in detail at the start of this section and represented in Lines 24-26 in Algorithm 3

Vocabulary density score s_v captures the importance of a candidate by analyzing the density of domain-specific words V around it. If a candidate c_i has more domain-specific words around it in a neighborhood radius of R, then it is considered more relevant to the domain. First, the distance from a candidate c_i to every vocabulary word $v_w \in V$ present in the text is calculated. Then scores to each candidate c_i are provided based on the density of vocabulary words around a candidate with in a neighborhood distance R. The neighborhood distance can be chosen based on the analysis of a document or can also be modified such that the vocabulary words v_w that occur in the same sentence as that of a candidate c_i are weighted more compared to other vocabulary words. If scores about the importance of a vocabulary word are available, then the vocabulary words v_w surrounding a candidate can be weighted accordingly. Lines 27-33 in Algorithm 3 describes this process.

$$s_v(c_i) = \sum_{v_w \in V} Score(c_i, v_w, R) \tag{3.11}$$

After obtaining all individual scores, a weighted average of all the scores is considered to obtained a combined final score for all the *where* candidates. The candidates are then ranked (sorted) based on their final location candidate scores (FS) to obtain the best candidate that closely corresponds to the answer for *where*.

$$FS(c_i) = \frac{\alpha_1.s_w + \alpha_2.s_s + \alpha_3.s_t + \alpha_4.s_g + \alpha_5.s_p + \alpha_6.s_v}{\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6} \tag{3.12}$$

3.3.2 Extraction of When

In this step, we derive an ordered set of date and time candidates that best represents the answer to the second Journalistic W, i.e., *when* in a article. The inputs to this step are a text document D, meta data associated with an article D_t, sentence number of the best *where* candidate D_w and returns the scores corresponding to every date/time mention in the article. The highest scored date/time mention is considered as *when* corresponding to an event.

Candidate identification task identifies candidates representing either date or time entities are obtained using a named entity recognition system or Stanford time tagger [21]. Additional temporal extraction techniques like syntactic rules can also be used for obtaining candidates. Information regarding these candidates such as the word number, sentence number in the document, and its resolution to common date format (YYYY-MM-DD) is obtained. The time tagger used in this process also converts relative time mentions (e.g. today) to absolute time mentions (a particular date value) based on the article publishing date which is provided as a parameter to the time tagger function.

The information about the potential candidates is used for scoring/ranking them using several factors such as its position in the document (word score) s_w, its difference to the reference date s_d, its distance from the best *where* candidate s_{wd}, and its frequency in the entire article s_f. Factors that are unique for this sub-step are discussed below:

Difference score s_d captures the importance of a candidate c_i to the actual occurrence of an event time. If an event occurs, then it is usually reported immediately. In general, the importance of a candidate decreases over time and this normalized score tries to capture the same. If a candidate c_i is more closer to the publishing date, then

Algorithm 4 Extraction of $When$

```
1: $D$: Document
2: $D_t$: Date of publishing.
3: $D_w$: Best *where* candidate (sentence number)
4: **procedure** $when$EXTRACTOR$(D, D_t, D_w)$
5:     $D' \leftarrow$ preprocess(D).
6:     $C \leftarrow \{\}$
7:     $Words : W \leftarrow wordTokenizer(D')$
8:     $Sentences : S \leftarrow sentTokenizer(D')$
9:     $N \leftarrow length(W)$
10:    **for** each $w_i \in W$ **do**
11:        **if** $NER(w_i)$ is $PERSON$ or $ORG$ **then**
12:            $C \cup \{\{w_i, i, resolve(w_i)\}\}$
13:        **end if**
14:    **end for**
15:    $Finalscores : FS \leftarrow \{\}$
16:    **for** each candidate $c_i \in C$ **do**
17:        $s_w \leftarrow 1 - \frac{i}{N}$
18:        $s_d \leftarrow datediff(c_i, D_t)$
19:        $s_{wd} \leftarrow c_i[2] - D_w$
20:        $s_f \leftarrow count(c_i[3])$
21:        $FS+ = \{c_i, whenscore(s_w, s_d, s_{wd}, s_f)\}$
22:    **end for**
23:    **return** $FS$
24: **end procedure**
```

it has a higher chance of being the actual *when* candidate. The difference between a date/time candidate c_i and the article's publishing date is calculated and represented as difference scores s_d. These scores are then resolved to absolute time s_d to capture the importance of a date/time candidate in an article.

$$s_d(c_i) = \frac{diff(c_i, rd)}{max_{c_i \in C} diff(c_i, rd)} \tag{3.13}$$

Where distance score [1]*:* s_{wd} is used for capturing the importance of a *when* candi-

[1]Distance to best *when* candidate can also be used in computing the final scores for *where*. Our approach mainly focuses on the spatio-temporal aspect of events. So, we consider distance to best *where* candidate for computing final scores to all the other Ws.

date to the best *where* candidate D_w. In general, all information related to an event tend to occur near one other. This score provides additional confidence to a candidate c_i if it occurs near to D_w. The number of sentences between a *when* candidate c_i and the best *where* candidate W (sentence number of best *where* candidate) is computed as s_{wd}.

$$s_{wd}(c_i) = \frac{diff(i, D_w)}{max_{c_i \in C} diff(i, W)} \tag{3.14}$$

Frequency score: s_f captures the number of occurrences of a resolved date/time candidate in the entire document D. If a word is mentioned multiple times in an article, then it carries more weights in the context of an article. This score captures the same, by providing higher scores to resolved candidates. All candidates are resolved to absolute time using the publishing date as the reference. Also, if a candidate cannot be parsed to a date format, then it is provided a lesser score [47].

$$s_f(c_i) = \frac{freq(c_i)}{max_{c_i \in C} freq(c_i)} \tag{3.15}$$

A weighted average of all the scores is considered to obtained a combined final score for all the *when* candidates. The candidates are then ranked (sorted) based on their final date/time scores (FS) to obtain the best candidate that closely corresponds to the answer for *when*.

$$FS(c_i) = \frac{\alpha_1.s_w + \alpha_2.s_d + \alpha_3.s_{wd} + \alpha_4.s_f}{\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4} \tag{3.16}$$

3.3.3 Extraction of Who and What

In this step, we derive the phrases which best represents the answers to the third and fourth Journalistic W's i.e., *who* and *what* in a article. The procedures for obtaining *who* and *what* candidates are merged together as there is a lot of interdependence between the two as mentioned in [47]. For example, a sentence 'Protesters gathered to submit an appeal to the government.' has both actors and the action mentioned in it. The candidates for *what* include certain action verb phrases (subject, object phrases) that represent the actual occurrence of an event. As most journalistic articles try to describe an event in the form who-did-what-to-whom [20], our model tries to leverage this syntactic structure to obtain candidates for *what* based on *who* candidates. Hence, we combine the extraction of *who* and *what*.

This step takes the text document D, a set of common name prefixes and suffixes, a list of preposition scores P_{who} and a list of domain-specific words V. The output of this sub-step are a list of ranked candidates for both *who* and *what* corresponding to an event in an article. The list of preposition scores are obtained using the Equation 3.6 described in Section 3.3.

Algorithm 5 Who and $What$ Candidate Identifier

```
1:  $D$: Document
2:  procedure $Who$ AND $What$ CANDIDATE IDENTIFIER($D$)
3:      $D' \leftarrow$ preprocess(D).
4:      $C_{who}, C_{what} \leftarrow \{\}$
5:      $Words : W \leftarrow wordTokenizer(D')$
6:      $Sentences : S \leftarrow sentTokenizer(D')$
7:      $N \leftarrow length(W)$
8:      for each $w_i \in W$ do
9:          if $NER(w_i)$ is $PER$ or $ORG$ then
10:             $C_{who} \cup \{\{w_i, i, s_i\}\}$
11:         end if
12:     end for
13:     for each $c_i \in C_{what}$ do
14:         if $root(deptree(c_i[3])) \in V$ then
15:             $C_{what} \cup \{\{rightsibling(deptree(c_i[3])), i\}\}$
16:         end if
17:     end for
18:     return $C_{who}, C_{what}$
19: end procedure
```

Algorithm 6 Extraction of Who and $What$

```
1: $D$: Document
2: $D_w$: Best *where* candidate (sentence number)
3: $T$: Title of the article
4: $P_{who}$: Preposition likelihood scores for *who*
5: $V$: Vocabulary (List of Domain-specific words)
6: $R$: Neighborhood radius
7: procedure $Who$ AND $What$EXTRACTOR$(D, D_w, T, P_{who}, V, R)$
8:     $D' \leftarrow$ preprocess(D).
9:     $C_{who}, C_{what} \leftarrow \{\}$
10:    $Words : W \leftarrow wordTokenizer(D')$
11:    $Sentences : S \leftarrow sentTokenizer(D')$
12:    $N \leftarrow length(W)$
13:    $C_{who}, C_{what} \leftarrow whoandWhatCandidateIdentifier(D)$        ▷ Algorithm 5
14:    $Finalscores : FS, FWS \leftarrow \{\}$
15:    for each candidate $c_i \in C_{who}$ do
16:        $s_w \leftarrow 1 - \frac{i}{N}$
17:        $s_s \leftarrow cosinesimilarity(S_i, T)$
18:        if $c_i \in T$ then
19:            $s_t \leftarrow 1$
20:        end if
21:        if $W[i-1] \in P_{who}$ then
22:            $s_p \leftarrow P_{who}[W[i-1]]$
23:        end if
24:        $Vwords \leftarrow \{\}$
25:        $s_{wd} \leftarrow diff(c_i[3], D_w)$
26:        for $r$ in $range(1, R)$ do
27:            if $W[i+r]$ or $W[i-r]$ in $V$ then
28:                $Vwords+ = \{W[i+r], r\}$
29:            end if
30:        end for
31:        $s_v \leftarrow Sum(Vwords)$        ▷ weighted sum function
32:        $FS+ = \{c_i, actorscores(s_w, s_s, s_t, s_p, s_v, s_{wd})\}$
33:    end for
34:    for each candidate $c_i \in C_{what}$ do
35:        $s_w \leftarrow 1 - \frac{c_i[2]}{N}$
36:        $s_s \leftarrow cosinesimilarity(S_{c_i[2]}, T)$
37:        $s_d \leftarrow diff(i, best(C_{who}), D_w)$
38:        $FWS+ = \{c_i, actionscores(s_w, s_d)\}$
39:    end for
40:    return $FS, FWS$
41: end procedure
```

The candidate identification task for *who* is the same as the candidate extraction process for *where* as described in Algorithm 5. However, instead of using locations as *where* we use *people* and *organizations* as entities in NER process. Our generic approach can be improved by using a curated list of prominent people, organization names can also be used to improve the results of this extraction process. A single event might contain multiple *who* candidates as opposed to *where* which contain only a single candidate. The scoring of the candidates for *who* is also exactly the same as *where* with the exception of gazette score. An additional score that obtains the distance of a *who* candidate to the best *where* candidate D_w is also includes while ranking the candidates for *who* as described in Section 3.3.2. The scores that are used for ranking *who* candidates are the word position score s_w, sentence similarity score s_s, title score s_t, preposition score s_p, vocabulary density score s_v and *where* distance score s_{wd} as discussed in Lines 15-30 in Algorithm 6.

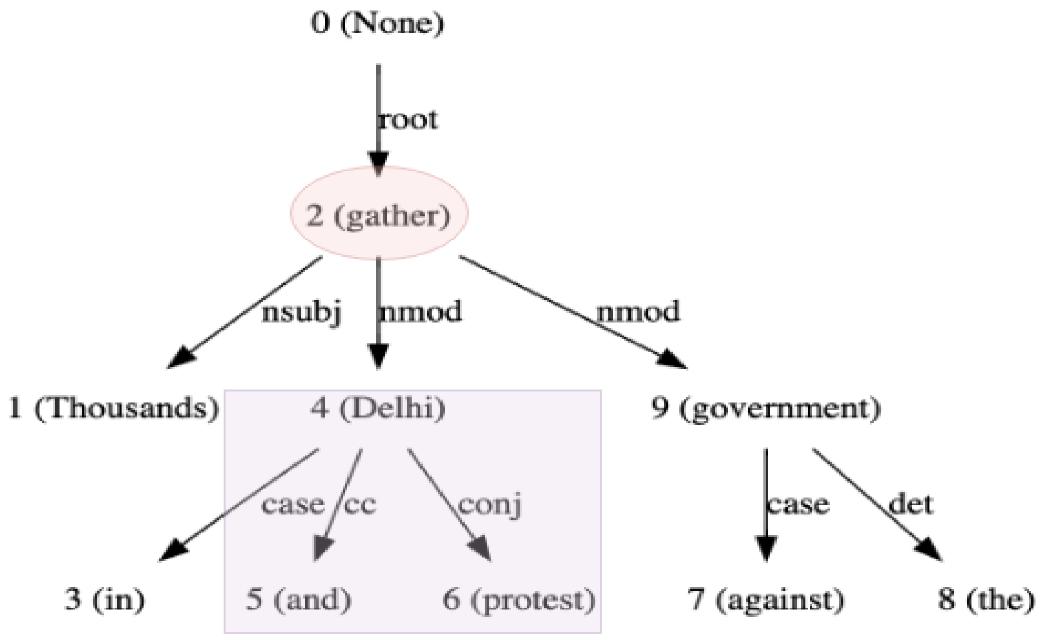


Figure 3.4: A sample dependency tree showing action related verb and *what* candidate.

Candidate identification task for *what* candidates uses a neural dependency parser to obtain the dependency tree for all the sentences that contain a *who* candidate as described in Lines 13-17 of Algorithm 5. These dependency trees are then parsed to obtain phrase candidates for *what* based on a domain-specific vocabulary V. If a dependency tree consists of an action verb similar to those in V, then the right sibling of that node including the verb is considered as a candidate for *what*. Figure 3.4 represents a sample dependency tree. Node2 represents the root of the tree which is an action verb. The right sibling of the verb node along with the verb is a candidate for *what*.

The candidates for *what* are ranked based on three different factors, its position in an article s_w, its similarity to the title s_s and its distance to the best *who* and *where* candidate s_d. The scoring mechanisms for these candidates are similar to those described for *where* and *when* extraction from Sections 3.3.1 and 3.3.2 and described in Lines 33-38 in Algorithm 6.

3.3.4 Extraction of Why

In this step, we extract the phrases that represents the answer to the final Journalistic W, i.e., *why* which corresponds to the reason for occurrence of an event if one exists in a document. This step considers a text document D, a set of action-related verbs [44, 6] and adverbs that best describe causal phrases C_v, best *where* candidate W and outputs a phrase that details the reason for occurrence of an event[47]. Obtaining the exact information for *what* and *why* is a challenging task as it depends on the linguistic features which can vary among various languages, this model can be easily extended for multiple languages by modifying the rules that analyze the dependency tree.

Why represents the reason for occurrence of an action. Not all articles might contain the candidates for *why* even if an event is mentioned in it. For example, if an event occurs it is reported in an article immediately even if the reason is unknown which might be available at a later time and the article is updated. The reason for an action is usually linked using causal verbs or conjunctions in a sentence. Our approach leverages a list of causal verbs, adverbs and conjunctions from [44] which are obtained using a data driven approach.

Algorithm 7 Extraction of Why

```
1: $D$: Document
2: $C_v$: List of causal verbs, adverbs, conjunctions.
3: $D_w$: Best $where$ candidate (sentence number)
4: procedure $Why$Extractor($D, C_v, D_w$)
5:     $D' \leftarrow$ preprocess(D).
6:     $C \leftarrow \{\}$
7:     $Words : W \leftarrow wordTokenizer(D')$
8:     $Sentences : S \leftarrow sentTokenizer(D')$
9:     $N \leftarrow length(W)$
10:    for each $s_i \in S$ do
11:        if $root(s_i) \in C_v$ then
12:            $C \cup \{\{Nounphrase(s_i), i\}\}$
13:        end if
14:    end for
15:    $Finalscores : FS \leftarrow \{\}$
16:    for each candidate $c_i \in C$ do
17:        $s_w \leftarrow 1 - \frac{i}{N}$
18:        $s_{wd} \leftarrow c_i[2] - D_w$
19:        $FS+ = \{c_i, Whyscore(s_w, s_{wd})\}$
20:    end for
21:    return $FS$
22: end procedure
```

First, all the phrases that either contain a verb (or verb form) from the Table 3.1 or adverbs from Table 3.2 are considered as candidates. These candidates are further filtered based on their syntactic structure: only candidates that have a noun phrase after the verb/adverb are retained. Some articles might not contain any of the verbs or adverbs from Tables 3.1 and 3.2 which indicates that the cause for an event is not explicitly mentioned in the article. Not all news articles report *why* related to an event in the title of an article. So, using sentence similarity score to title doesn't contribute to ranking *why* candidates. Also, preposition scores are ignored because the candidates for *why* are often associated with causal verbs or conjunctions but not prepositions.

The position scores s_w and the distance to best *where* candidates s_{wd} as described in Sections 3.3.1 and 3.3.2 are used for computing the final *why* scores. This *why* extractor might also consider weighted scores for various verbs and adverbs to better rank the candidates.

Table 3.1: A list of causation verbs [44]

Causal Verbs			
give rise (to)	stir up	create	start
induce	entail	launch	make
produce	contribute (to)	develop	begin
generate	setup	bring	rise
effect	trigger off	stimulate	
bring about	commence	call forth	
provoke	set off	unleash	
arouse	set in motion	effectuate	
elicit	bring on	kick up	
lead (to)	conduce (to)	give birth (to)	
trigger	educe		
derive (from)	originate in	call down	
associate (with)	lead off	put forward	
relate (to)	spark	cause	
link (to)	spark off		
stem (from)	evoke		
originate	link up		
bring forth	implicate (in)		
lead up	activate		
trigger off	actuate		
bring on	kindle		
result (from)	fire up		

Table 3.2: A list of adverbial indicators and causal conjunctions [6]

Adverbial Indicators	Causal Conjunctions
therefore	consequence (of)
hence	effect (of)
thus	result (of)
consequently	outcome (of)
accordingly	stemmed (from)
	because
	hence
	stemmed
	due

These candidates are ranked based on factors like sentence position score as described in Section 3.3.1 and its distance to best *where* candidate as described in Section 3.3.2. If weights for causal verbs/adverbs are available, then they can also be utilized in computing the final score.

Chapter 4

Implementation and Results

The 5WE and associated algorithms are implemented using Python and requires minimal human intervention. The inputs to these algorithms are textual data in English and its corresponding meta data. These set of algorithms depend on various other packages available for python such as beautifulsoup [99], tensorflow [1], flair [4], nltk [72], corenlp [77], fuzzywuzzy [45], various pre-trained neural network models and pandas[80]. The corenlp framework [77] should be started separately in a terminal and appropriate connections must be made in the Python program so as to use the co-reference system and SU-time.

4.1 Datasets

Global Database of Events, Language and Tone (GDELT) [68] is a database of global human society which monitors world's broadcast, print and web news from nearly every corner in over 100 languages. Tracing events back to 1979, GDELT database utilizes 20 categories to define events including, but not limited to, protests, threats and uses of unconventional mass violence [11]. The selected categories for this research are: Appeal, Demand, Threaten, Coerce, Protest, Assault, Fight and Engage in Unconventional Mass Violence (UMV).

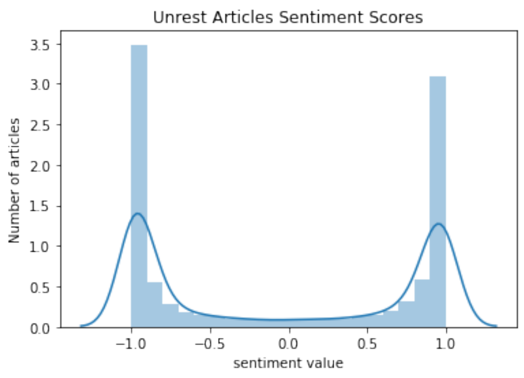
The geographic scope of this thesis is limited to India for the year 2017. The GDELT events from January, 2017 to December, 2017 corresponding to the categories listed above from India (IN) are collected and filtered. These events are then filtered based on the URL field, such that if an URL consists of either Times of India or Indian Express [55, 109], then those events are considered. Any duplicate mentions of the same event are discarded based on the date and location provided by GDELT. However, one drawback of this approach is that multiple unrelated events that are reported on the same date and location are considered as duplicates and hence discarded.

The URLs of the cleaned dataset are used for downloading the content, meta data of news articles corresponding to that event using beautifulsoup[99] in Python. If a URL is unresponsive or the data returned is null, then that event is dropped from the dataset. A final social unrest (52,000) articles dataset consisting of the latitude, longitude of the event provided by GDELT, date, content of the news article and its category is prepared.

4.1.1 Classification Dataset

Since classification require other articles which are not related to social unrest, a dataset consisting of (360,000) articles related to sports, finance, technology and travel are downloaded from BBC news [12] and various other sources. Any article which consists of fewer than 7 lines of data is discarded resulting in a balanced, combined dataset of 87,500 articles made up of both social unrest and other articles. This dataset is further divided into training and test data using an 80-20 split. All of the models are trained using 10-fold cross validation.

The sentiment analysis task is used to obtain the attitude of an article to a social unrest event. Most sentiment analysis tools consider the syntactic structure of a



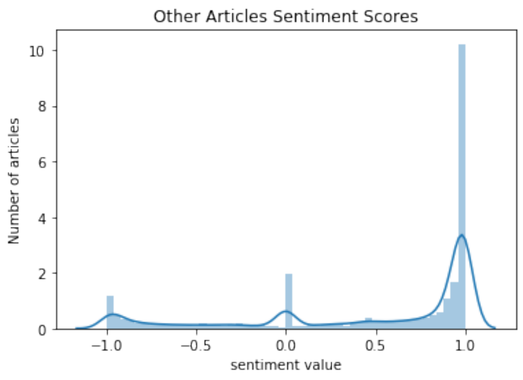


Figure 4.1: Sentiment scores visualization

sentence to obtain the sentiment. Some, recent deep learning techniques consider the semantic *relation* among words to capture the sentiment of a document. We use vader sentiment analysis and textblob tools to obtain the sentiment of a document which use the syntactic structure of a sentence and some pre-defined sentiment scores for certain words to obtain the overall sentiment score for a document. We modify and add some sentiment scores for social unrest keywords to the vader sentiment tool [54]. Figure 4.1 represents the distributions of sentiment scores for both social unrest and other articles. Figure 4.2 represent the heat map obtained by plotting the sentiments of all social unrest related articles.

4.1.2 Evaluation Dataset

A standardized dataset for 5Ws corresponding to textual data is not readily available which poses a significant challenge in our research. A 5Ws evaluation dataset is manually prepared with the help of 6 human coders. In Phase 1, a group of 10 social unrest related documents are given to all 6 coders so as to gain familiarity with social unrest related documents. Additional information about common city names, acronyms, major organizations, geographical and demographic structure about Indian sub-continent is provided to help in the annotating process.

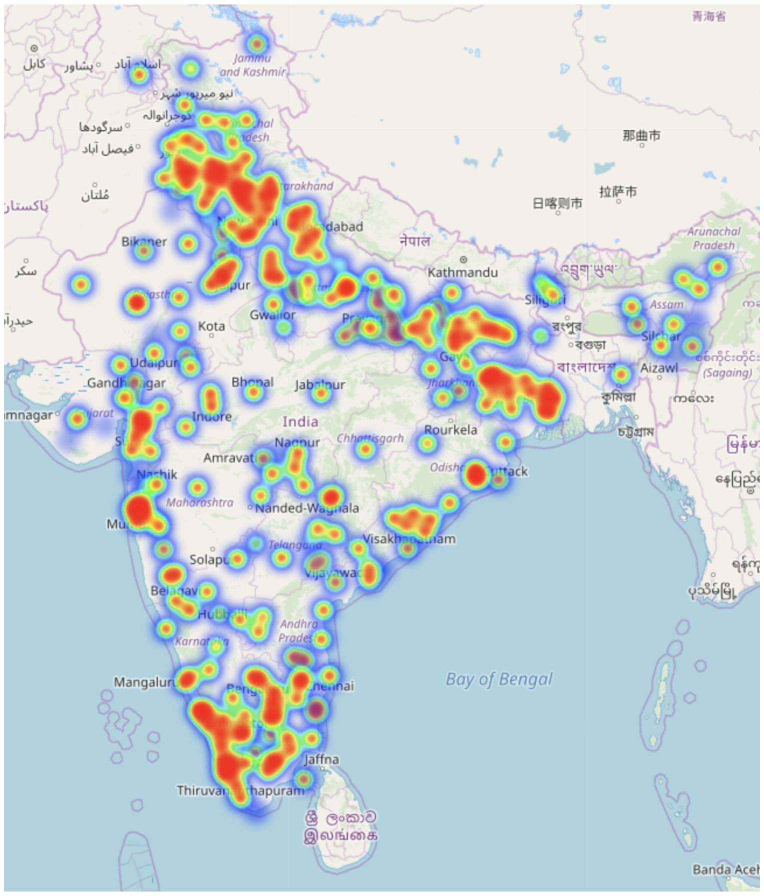


Figure 4.2: Visualization of social unrest related articles

The annotations for these 10 sample documents are collected and analyzed manually such that the annotations match among all the coders. If any ambiguous cases exist, then they are discussed so as to either include all the annotations or ignore some of them that aren't agreed by majority of the coders. For example, if a long sentence is annotated as *why*, then it is analyzed further to include only the corresponding cause/reason. The intercoder reliability (ICR) [52] metric among different coders in Phase 1 is presented in Table 4.1. This metric computes the reliability measure of the human coders by observing the number of annotations that overlap among two coders. One coder during Phase 1 didn't annotate most of the answers to 5Ws. So, we removed that coder and computed the ICR values as presented in Table 4.2.

Table 4.1: ICR measures among human coders in Phase 1

W	*Minimum ICR*	*Maximum ICR*	*Average ICR*
Where	0.22	0.76	0.43
When	0.00	0.87	0.31
Who	0.07	0.57	0.28
What	0.14	0.58	0.31
Why	0.00	0.26	0.13
All Ws			**0.29**

Table 4.2: ICR measures among 5 human coders in Phase 1

W	*Minimum ICR*	*Maximum ICR*	*Average ICR*
Where	0.28	0.76	0.46
When	0.00	0.87	0.35
Who	0.10	0.57	0.31
What	0.14	0.58	0.33
Why	0.00	0.26	0.14
All Ws			**0.32**

We can observe that the minimum ICR for *when* and *why* are 0. The low ICR values can be attributed to ambiguous phrases in the document. Also, the ICR measure used here considers only perfect match among various annotations. For example, a location *Chennai city* is annotated by coder A and *Chennai* is annotated by coder B. Even though both of them relate to the same location, they are considered as different annotations. These comparisons impact the ICR computation of *what* and *why* scores as one coder might annotate an entire phrase P_1 for *why*, whereas other coder might annotate only a sub-phrase with in the phrase P_1. Adjusted ICR values are computed by considering these cases and is presented in Tables 4.3 and 4.4.

After all coders agree on the annotations for these sample documents. A single set of 100 documents is sampled from a large (500) unrest related news articles dataset. This single set is assigned to two human coders. However, we were able to obtain

Table 4.3: Adjusted ICR measures among human coders in Phase 1

W	*Minimum ICR*	*Maximum ICR*	*Average ICR*
Where	0.22	0.87	0.51
When	0.18	0.94	0.48
Who	0.21	0.78	0.50
What	0.20	0.64	0.38
Why	0.00	0.89	0.35
All Ws			**0.44**

Table 4.4: Adjusted ICR measures among 5 human coders Phase 1

W	*Minimum ICR*	*Maximum ICR*	*Average ICR*
Where	0.32	0.87	0.56
When	0.18	0.94	0.62
Who	0.23	0.78	0.56
What	0.20	0.64	0.39
Why	0.11	0.89	0.37
All Ws			**0.50**

annotations for only 74 documents. The coders annotate the *best* 5Ws corresponding to an event in the news article using Dataturks [112] as shown in Figure 3.2. The ICR values for the final annotated dataset consiting of 74 documents by 2 coders are presented in Table 4.5. The adjusted ICR has a value of 0.75 which asserts the robustness of ground truth.

Table 4.5: ICR measures for documents in Phase 2

W	*Traditional ICR*	*Adjusted ICR*
Where	0.38	0.74
When	0.51	0.83
Who	0.42	0.76
What	0.41	0.76
Why	0.38	0.66
All Ws	**0.42**	**0.75**

4.2 Results

The results for each of the steps in the 5Ws extraction pipeline are presented in this section. First, accuracies of various machine learning and deep learning approaches on social unrest related news articles dataset is described in Section 4.2.1. Section 4.2.2 describes about the results for individual Ws on the manually annotated dataset.

4.2.1 Document Classification Results

Several experiments were ran using various machine learning algorithms on the balanced dataset. First, all stop words are removed and the words in a document are converted into vector representations using pre-trained Word2Vec or GloVe embeddings (300 dimensions). These embeddings are then converted to a fixed length for the entire document by either averaging them or considering the minumum or maximum of those embeddings. TF-IDF vectors that are generated based on a corpus of documents are also used for representing words. Averaging all word embeddings provides a better representation for the entire document compared to min/max representations. Table 4.6 represents the classification accuracies of various machine learning algorithms with different embeddings. The underlined values represent the best classification accuracies using a specific algorithm and the bold value represent the best classification accuracy among all values.

Table 4.6: Classification accuracies using 10-fold cross validation

Classifier Name	TF-IDF (%)	word2vec (avg)(%)	GloVe (avg)(%)	word2vec (min+max)(%)	GloVe (min+max(%))
Logistic Regression	<u>**85.55**</u>	78.43	74.08	78.85	75.41
Multinomial Naive Bayes	<u>75.57</u>	69.95	63.36	70.10	62.30
Random Forest (n=200)	70.29	<u>73.45</u>	71.71	73.20	73.10
Support Vector Machine	<u>75.34</u>	74.40	73.16	75.24	73.38

In general, the classification methods performed better while using Word2Vec embeddings. Logistic regression using TF-IDF embeddings has the highest classification accuracy compared to all other classifiers. Representing the entire document using either averaging of word embeddings or combining minimum and maximum didn't impact the classification results.

Next, we explore a different type of embedding which is obtained by using the entire document called Doc2Vec embeddings [66]. A Doc2Vec model from gensim [98] which is already trained on a large dataset of IMDB paragraphs is used to obtain embeddings for every article. Table 4.7 represents the accuracies obtained using Doc2Vec embeddings of size 100 and 1000 on machine learning classifiers. Changing the size of embeddings had a little impact on the classification accuracies. All of these methods were also tried using sentiment value of the entire document as an additional feature in the embeddings, which had no impact on the classification results.

Table 4.7: Classification accuracies using Doc2Vec embeddings of 100 and 1000 dimensions.

Classifier Name	Doc2Vec (100)(%)	Doc2Vec (1000)(%)
Logistic Regression	73.50	**73.70**
Multinomial Naive Bayes	59.10	60.80
Random Forest (n=200)	68.74	68.67
Support Vector Machine	68.80	68.17

To improve the accuracies, we tried ensemble methods where different models are combined so as to learn from the mistakes made by previous classifiers. The output of these classifiers are combined using majority voting scheme and trained using GloVe (avg) embeddings as these seem to perform better compared to other embeddings. An additional stacking approach consisting of K-nearest neighbors, XGBoost and Extra trees classifer with logistic regression as meta classifier is also tested on the balanced dataset. A majority voting ensemble method of logistic regreesion, extra trees and

SVM is also considered. The results of these boosting and additional approaches is presented in Table 4.8.

Table 4.8: Classification accuracies using ensemble methods

Classifier Name	GloVe (avg) (%)
Extra Trees Classifier[42]	79.76
Adaboost (100 trees)[49]	79.07
Xgboost[23]	79.39
K-nearest neighbors[30]	79.39
Majority voting [93]	78.90
Stacked[46]	**80.57**

We can observe minor improvements in accuracies compared to single machine learning algorithms. K-Nearest Neighbors (KNN) approach is a simple algorithm but performs significantly better compared to various boosting approaches. Stacked classifer has the highest classification accuracy among all ensemble methods. Even though the boosting approaches performed better to traditional machine learning approaches, these methods require additional computational resources and time on a large dataset.

Fasttext [59] classification on the balanced dataset produced better accuracies compared to previous methods. First, the dataset is divided into training and test sets consisting of 60,000 and 8,000 articles, respectively. Several experiments with varying regularization parameter values, loss functions and test sets are evaluated while training for 25 epochs. Test sets containing social unrest related keywords and without keywords were considered for evaluating the impact of the presence of those keywords in classification process. Table 4.9 presents the various results and it can be observed that this model performs better even in the absence of social unrest related keywords.

Table 4.9: Classification accuracies using fasttext classifier

Model	Classification accuracy (%)
Softmax loss, ngrams size 1	81.94
Softmax loss, ngrams size 2	90.21
Hierarchical loss, ngram size 1	99.95
Hierarchical loss, ngram size 2	**99.98**
Hierarchical loss, test set with no keywords	99.63

Recurrent neural network architectures with and without attention mechanisms were also trained for 10 epochs and tested using Keras framework. Simple bi-directional neural network with a single layer attention is trained using adam optimizer on a subset of data consisting of 18k articles. Several other models with two layers of attention, one at the word level and other at sentence level were also used with both LSTM and GRU type recurrent units. A final model with Bi-directional LSTM units with hierarchical attention is trained on a larger dataset of 70,000 articles and tested on 18,000 articles, *Bi-LSTM2*, the results of all these models are presented in Table 4.10.

Table 4.10: Classification accuracies using recurrent neural network architectures

Model	Classification accuracy (%)
Bi-LSTM (single attention)	84.63
Bi-GRU (hierarchical attention)	88.20
Bi-LSTM (hierarchical attention)	88.35
Bi-LSTM2 (hierarchical attention)	**99.55**

Figure 4.3 represents the loss graphs for a Bi-LSTM network with single attention and hierarchical attention. We can observe that using hierarchical attention provides better accuracies as it analyses the document by sentence structure which takes important words into consideration. Similarly, using a large dataset significantly increases the classification accuracy value for this architecture.

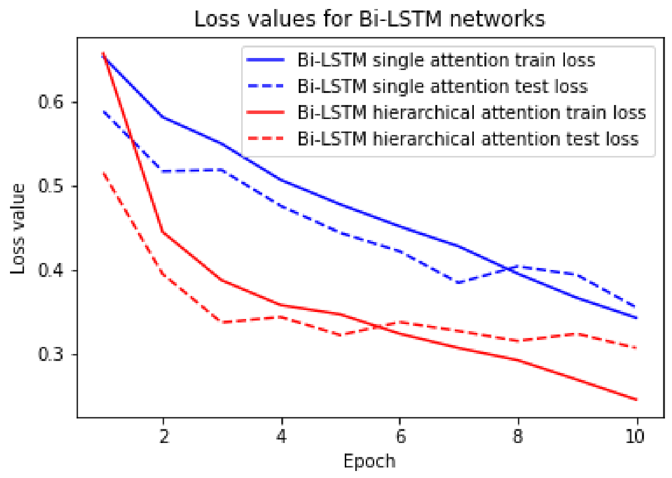


Figure 4.3: Bi-LSTM network loss values

The classification of social unrest related documents is almost perfect using Fasttext classifier. After evaluating several classification techniques, we can say that models which consider the contextual information rather than traditional approaches perform better. The embeddings obtained at the final layer of either Fasttext or the Bi-LSTM models can be used further in the 5W extraction process. For example, an additional similarity score which calculates the cosine similarity between a sentence and the obtained document embeddings from these processes can be utilized in the candidate ranking task. Overall, the classification processes discussed in this section can be incorporated in to the 5Ws extraction framework.

4.2.2 Evaluation of the 5WE approach

The results obtained by the 5Ws extraction process are described below. Since, most of these algorithms depend on the usage of preposition scores obtained from a large set of documents. We describe the process of obtaining those scores and present the distributions of those scores. Then we describe the annotation dataset and the evaluation methods used for obtaining accuracies. We also discuss the failures/drawbacks of our 5Ws extraction process. Finally, we present the correlation among individual scores that are used for calculating final scores.

Preposition scores are used in *where* and *who* extraction processes. These scores are extracted using a large social unrest related news articles dataset consisting of 24000 articles using the approach described in Section 3.3. The prepositions ordered by their likelihood scores are then utilized in scoring the candidates. The prepositions for *where* and *who* are obtained independently and their distributions are presented in Figures 4.4 and 4.5. This data driven approach can be easily tailored for other domains like human rights violation by considering articles related to that specific domain.

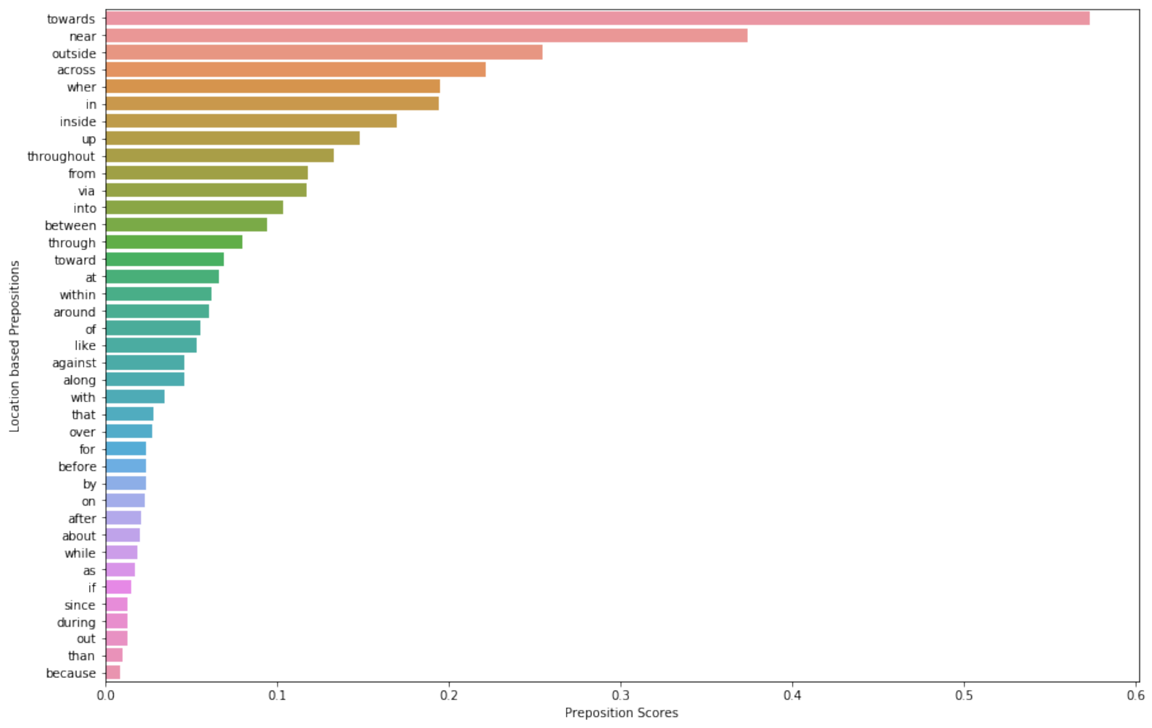


Figure 4.4: Preposition weights associated with locations.

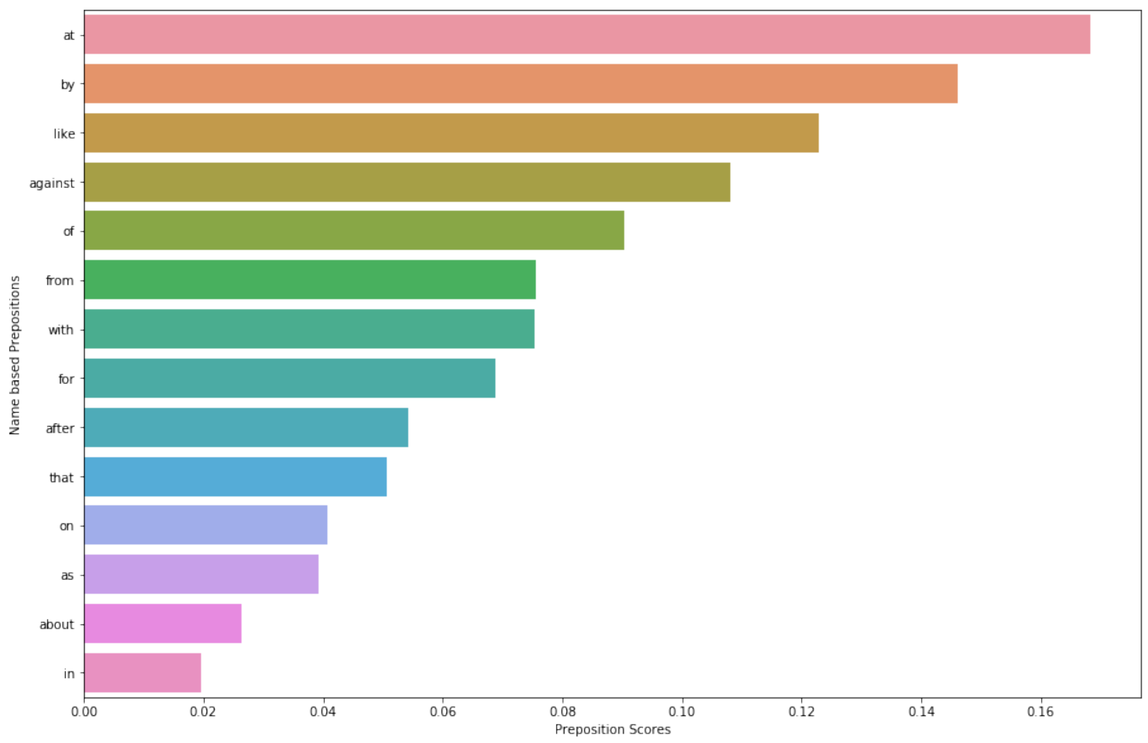


Figure 4.5: Preposition weights associated with *who* candidates

Also, the number of occurrences of the prepositions for *where* was much lower compared to *who*. So, even though *towards* appears less frequently, it is strongly associated with a location candidate whenever it occurs. But the same cannot be said about *at* for *who* candidates as the likelihood of *at* before a *name* or *organization* is much lower.

In this study, we evaluated 74 manually annotated news articles for 5Ws by two human coders whose ICR values are presented in Table 4.5. All of the news articles were annotated by at least 2 human coders. If the two coders agree on a W annotation, then it is considered as ground truth. If they do not agree, then the union of annotations by both coders is considered as ground truth. Some news articles doesn't contain any annotation for *why* and the ground truth for those articles is given as *nan* (not available). All of these articles were given as inputs to the 5Ws extraction processes and evaluated using three evaluation methods as discussed below.

- *Exact Match:* The candidate with the highest score produced by the 5Ws extraction processes should match the ground truth annotation.
- *Top-3 Match:* The ground truth annotation for a *W* is present in the set of top-3 candidates ordered by scores.
- *Candidate Check:* The ground truth annotation is present in the set of all extracted candidates. It captures the accuracy of the candidate identification task.

The accuracies obtained for each individual 5Ws extraction process using each of the three evaluation methods are presented in Table 4.11. The numbers in the parenthesis represent the number of ground truth annotations for 74 documents. To better improve the accuracies two additional verbs (demand, against) are included for extracting *why*. Extracting candidates for *why* is significantly challenging compared to other *W* candidates as an article might or might not contain a reason/cause mentioned in it. Our approach was able to extract *why* candidates for only 43 articles and return *nan* for remaining articles. It is even challenging for human annotators for identifying *why* as different articles might report it using different syntactic styles.

Table 4.11: Accuracies of the 5Ws extraction processes

W	Exact Match (%)	Top-3 Match (%)	Candidate Check (%)
Where (139)	68.9	82.4	85.1
When (98)	71.6	91.8	91.8
Who (144)	48.6	74.3	97.2
What (84)	67.5	83.8	89.1
Why (61)	32.4	32.4	33.8
All 5Ws	57.8	72.9	79.4

The candidate identification task was successful in all Ws expect *why*. Next, we report the accuracies obtained by considering only the articles for which at least one candidate is generated for every W (42 articles) in Table 4.12. The number of ground truth annotations for these 42 articles are presented in the parenthesis in Table 4.12. We can observe that the accuracy for *why* increases if at least one candidate is produced.

Table 4.12: Accuracies of the 5Ws extraction processes if at least one candidate is extracted for every W.

W	Exact Match (%)	Top-3 Match (%)	Candidate check (%)
Where (67)	61.9	80.9	85.7
When (56)	78.5	88.1	88.1
Who (82)	40.5	64.3	95.2
What (57)	80.9	80.9	85.7
Why (48)	57.1	57.1	59.8
All 5Ws	63.8	74.3	82.9

Comparing Tables 4.11 and 4.12, we can observe that the overall accuracy of 5WE increases if at least one candidate is extracted for every W. There is a slight improvement in accuracy for *where* and a significant improvement for *why*. However, the accuracies for *when*, *who* and *what* decrease slightly. 5WE was able to extract candidates with high accuracy and was also able to rank candidates with comparable accuracy to the ICR value. The top-3 match and the candidate check strategies were surprisingly better than our human coders.

However, comparing the performance of this approach to existing approaches is not feasible because of the lack of a publicly available dataset. Other proposed approaches have either used a non-disclosed dataset or conducted experiments only on a specific W. The Giveme5W approach [47] obtains an average precision of 0.70 which is equal to the top-3 match accuracy value by 5WE.

The results for *when* and *what* that have either simple phrases as answers or a well defined structure have higher accuracies. The accuracies for *where* and *who* are lower compared to *when* and *what* because, the candidates for *where* and *who* are proper nouns which are specific to a particular geographical region. For example, *Ex-MLA* should be candidate for *who*, but is not captured by the *who* extraction process as it isn't included in the list of persons/organizations. Several other person names in India are similar to certain location names which aren't captured by the NER process.

4.3 Additional Considerations Regarding 5Ws

5WE performed better for all *Ws* except *why* as shown in Table 4.11. The accuracies of 5WE depend on the performance of different natural language processing techniques utilized in the process. The *where* extraction algorithm was able to extract the candidates but wasn't able to rank them according to their importance to the article which might be due to the absence of specific locations in the gazetteer. For example, a small street named *Hydernagar* has been extracted but wasn't present in the gazetteer, then it receives a lower score compared to other candidates. Another failure case, might be where the algorithm failed to extract a candidate because it is annotated as *other* by the NER. This process can be improved by using a better NER system that is specifically trained on location from a certain region, and using an extensive list of locations as gazetteer.

Table 4.13 presents the correlation among the individual scores used for obtaining a final *where* score. We can observe that position score and sentence score are closely related. This analysis can be useful for ignoring some of the scores or providing appropriate weights to individual scores in the computation of final score. The preposition scores that were used in the *where* extraction process had little impact on the final score because the scores for most of the prepositions are less than 0.4 which can be observed in Figure 4.4. Also, when the evaluation dataset is manually observed, the prepositions preceding location names have very low scores.

Table 4.13: Correlation among individual scores used for computing *where* final score

Scores							
	position	sentence	gazettee	title	unrest	preposition	final
position	**1**	-0.0373	**0.5708**	0.0445	0.1052	-0.0505	**0.7275**
sentence	-0.0373	**1**	-0.0562	**0.7859**	0.1017	0.0276	0.2993
gazetttee	**0.5708**	-0.0562	**1**	0.1490	0.0374	0.0733	**0.8079**
title	0.0445	**0.7859**	0.1490	**1**	0.0503	-0.0316	0.4697
unrest	0.1052	0.1017	0.0374	0.0503	**1**	-0.3119	0.3614
preposition	-0.0505	0.0276	0.0733	-0.0316	-0.3119	**1**	0.0632
final	**0.7275**	0.2993	**0.8079**	0.4697	0.3641	0.0632	**1**

The *when* extraction algorithm performs better compared to all the other approaches based on the results from Tables 4.11 and 4.12 because, an article usually contains very few candidates for *when* which are represented using a standard format. The SU-Time was able to extract almost all candidates present in an article. The failure cases for this method might include extracting locality specific event dates like *state formation day*, *state specific festival day* etc. Additional lists consisting of locality-specific dates can be provided to the algorithm to better improve its accuracy. Table 4.14 represents the correlation among the scores used for computing the final *when* score. We can observe that the date difference score and the distance to *where* score have high correlation compared to the frequency scores. As a news article usually mentions a date just once, the frequency score didn't help much in computing the final score.

Table 4.14: Correlation among individual scores used for computing *when* final score

Scores				
	date difference	frequency	distance to *where*	final
date difference	**1**	-0.0461	0.1790	**0.7930**
frequency	-0.0461	**1**	-0.1827	-0.1297
distance to *where*	0.1790	-0.1827	**1**	**0.7411**
final	**0.7930**	-0.1297	**0.7411**	**1**

The *who* extraction algorithm was able to identify candidates with high accuracy but failed to rank them based on their importance in a document and can be viewed in the Table 4.11. This failure can be attributed to the limitations of the co-reference resolution system used. Since, an article usually contain more proper nouns corresponding to either people or organization, it is common for a co-reference resolution system to make more mistakes in distinguishing the pronouns as shown in Figures 4.6 and 4.7. This failure in turn affects the frequency and the domain-specific scores used for computing the final scores. The failure cases for this method includes extracting common designated person names like *Governor*, *Manager*, *Leader of a Corporation* etc., as the NER system annotates these words as *other* category. Additional list of person names, designations can be used for improving the results of *who*. Table 4.15 includes the correlation among different individual scores used in computing the final score for *who*.

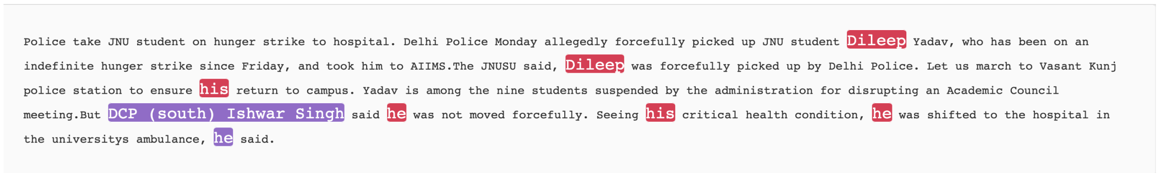
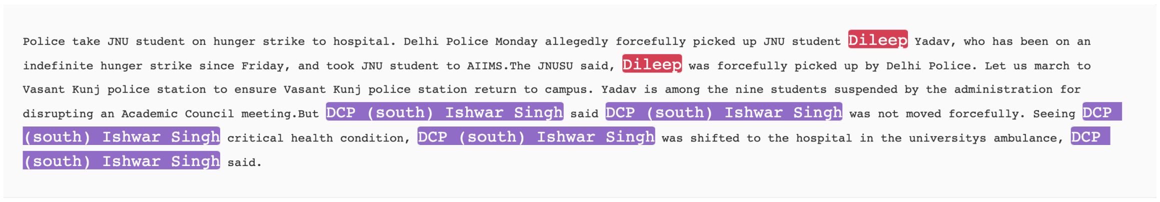

Police take JNU student on hunger strike to hospital. Delhi Police Monday allegedly forcefully picked up JNU student Dileep Yadav, who has been on an indefinite hunger strike since Friday, and took him to AIIMS.The JNUSU said, Dileep was forcefully picked up by Delhi Police. Let us march to Vasant Kunj police station to ensure his return to campus. Yadav is among the nine students suspended by the administration for disrupting an Academic Council meeting.But DCP (south) Ishwar Singh said he was not moved forcefully. Seeing his critical health condition, he was shifted to the hospital in the universitys ambulance, he said.

Figure 4.6: Correct coreference resolution



```
Police take JNU student on hunger strike to hospital. Delhi Police Monday allegedly forcefully picked up JNU student Dileep Yadav, who has been on an
indefinite hunger strike since Friday, and took JNU student to AIIMS.The JNUSU said, Dileep was forcefully picked up by Delhi Police. Let us march to
Vasant Kunj police station to ensure Vasant Kunj police station return to campus. Yadav is among the nine students suspended by the administration for
disrupting an Academic Council meeting.But DCP (south) Ishwar Singh said DCP (south) Ishwar Singh was not moved forcefully. Seeing DCP
(south) Ishwar Singh critical health condition, DCP (south) Ishwar Singh was shifted to the hospital in the universitys ambulance, DCP
(south) Ishwar Singh said.
```

Figure 4.7: Coreference resolution by CorefAnnotator [26]

Title score was not more useful in computing the final score for *who* candidates. Also, an interesting observation is that preposition scores were more useful for ranking *who* candidates but not for *where* candidates as shown in Table 4.12. Also, unrest scores were more useful for ranking *who* candidates as sentences are usually mentioned in the form *who-did-what-to-whom*. If the *what* phrase contains an unrest related word, then the associated *who* candidate receives higher unrest score. Another important thing to note is that even though unrest scores are highly correlated with final scores, the accuracy for *who* using exact match strategy is less as shown in Tables 4.11 and 4.12. This means that unrest scores might have negatively impacted ranking *who* candidates.

Table 4.15: Correlation among individual scores used for computing *who* final score

	Scores						
	position	sentence	frequency	title	unrest	preposition	final
position	**1**	0.0403	0.0100	-0.0679	0.0404	-0.2162	**0.5388**
sentence	0.0403	**1**	0.1479	-0.0137	**1**	-0.0960	**0.6448**
frequency	0.0100	0.1479	**1**	-0.0322	0.1479	-0.0467	0.3654
title	-0.0679	-0.0137	-0.0322	**1**	-0.0137	0.0872	-0.0154
unrest	0.0403	**1**	0.1479	-0.0137	**1**	-0.0960	**0.6448**
preposition	-0.2162	-0.0960	-0.0467	0.0872	-0.0960	**1**	0.3037
final	**0.5388**	**0.6448**	0.3654	-0.0154	**0.6448**	0.3037	**1**

The *what* extraction algorithm performs reasonably better compared to *who* extraction. Even though the extraction process for *what* is dependent on *who*, it obtained better results because of the syntactic structure of sentences in an article. For example, if a sentence consists of two actors and an action phrase. The *who* extraction algorithm might provide higher score to just one actor and hence receive lower exact match accuracy, but the *what* algorithm considers the entire action phrase which matches with the ground truth. Since the action phrases are matched based on the presence of a vocabulary related word in them, the algorithm is dependent on the vocabulary. For example, if a phrase *'sought asylum at local church'* is annotated as ground truth and the verb *seek* is not present in the domain-specific vocabulary, then the algorithm fails to extract the correct candidate. This process can be improved by using a sophisticated vocabulary with a better dependency parser.

Table 4.16 includes the correlation among different individual scores used in computing the final score for *what*. We can observe the high correlation between the distance to best *where* and *who* candidates and the final scores as important information tend to occur closely in a document. Also, since our *what* extraction approach considers the verb phrase around *who* candidates, the distance score is more correlated.

Table 4.16: Correlation among individual scores used for computing *what* final score

Scores				
	similarity	distance to *where* and *who*	position	final
similarity	**1**	0.1297	0.3739	0.3089
distance to *where* and *who*	0.1296	**1**	0.3011	**0.9812**
position	0.3739	0.3011	**1**	0.4110
Final	0.3089	**0.9812**	0.4110	**1**

Lastly, the *why* extraction algorithm performs worse compared to all the remaining *W* extraction approaches because extracting causal relationships from text itself is an arduous task. Using additional list of causal verbs and conjunctions can improve the accuracies of this extraction process. Identifying candidates for *why* depends on the context of an event mentioned in an article and not only on the syntactic structure of a sentence. Table 4.17 includes the correlation among different individual scores used in computing the final score for *why*. Since only 2 scores are used for computing the final score for *why*, both of them are highly correlated. As most of the information tends to occur at the start of text, all of the 5Ws scores are either moderately or highly correlated to their final scores.

Table 4.17: Correlation among individual scores used for computing *why* final score

Scores			
	position	distance to *where*	final
position	**1**	**0.9477**	**0.9926**
distance to *where*	**0.9477**	**1**	**0.9793**
Final	**0.9926**	**0.9793**	**1**

Overall, the 5WE performed well in extracting the answers to Journalistic 5Ws with accuracies in the range 63% to 83% as shown in Table 4.12. Furthermore, most of the component scores have been shown to be useful in ranking the candidates.

Chapter 5

Conclusion and Future Work

Extracting Journalistic 5Ws from text is important for understanding many social processes. In this thesis, we focused on developing a generalized framework for automatically extracting 5Ws using syntactic and semantic cues present in text. First, a classifier to identify articles related to a domain is developed. Several machine learning and deep learning algorithms are evaluated on a dataset consisting of social unrest articles.The impact of domain specific keywords in the process of classification using fasttext is discussed in Section 4.2.1.

The domain specific articles are then analyzed through a set of algorithms for extracting 5Ws based on heuristics that leverage specific words and their semantic representations. Challenges associated with the extraction of each W and the different principles used for ranking the 5W candidates are discussed in Section 3.3. Extending this generalized approach to a specific domain is studied in detail using a manually annotated 5Ws dataset of social unrest related articles. The entire 5W extraction approach is evaluated using 3 strategies and the results obtained by 5WE are closer to the ICR values by human coders. We further discuss the correlation among the individual scores used in computation of final scores. The drawbacks related to these approaches and ways to improve them in Section 4.3.

This approach can be improved using better NER, coreference resolution systems.

Further analysis about the individual scores can be conducted to determine the optimal set of weights for computing final scores. The potential future research directions include:

- These results can be used for clustering news articles based on 5Ws context and also for linking events across multiple sources. The *where* extraction process can be extended to obtain geographic coordinates which can be used for visualizing all related events on a map.
- The obtained 5Ws can be combined with socio-demographic features to simulate and forecast future events.
- The same approach can be customized to extract event details from social media like tweets, posts across multiple languages to perform real-time analysis.
- The 5WE can be further extended to include the event descriptor *how*.
- Developing an integrated 5WE using confidence measures for ranking a W candidates based on other Ws. For example, in this process we use the distance to *where* candidate as an individual score in all the subsequent W extraction processes. This can be made an iterative process so as to use all other W scores while scoring a specific W.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] David Ahn. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8, 2006.

[3] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[4] Zalando research Akbik. Flair. `https://github.com/zalandoresearch/flair/`, 2018.

[5] Felipe Almeida and Geraldo Xexéo. Word embeddings: A survey. *arXiv preprint arXiv:1901.09069*, 2019.

[6] Bengt Altenberg. Causal linking in spoken and written english. *Studia linguistica*, 38(1):20–69, 1984.

[7] Nabiha Asghar. Automatic extraction of causal relations from natural language texts: a comprehensive survey. *arXiv preprint arXiv:1605.07895*, 2016.

[8] Ben Athiwaratkun, Andrew Gordon Wilson, and Anima Anandkumar. Probabilistic fasttext for multi-sense word embeddings. *arXiv preprint arXiv:1806.02901*, 2018.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[10] Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. *Advances in automatic text summarization*, pages 111–121, 1999.

[11] Sudeep Basnet, Leen-Kiat Soh, Ashok Samal, and Deepti Joshi. Analysis of multifactorial social unrest events with spatio-temporal k-dimensional tree-based dbscan. In *Proceedings of the 2nd ACM SIGSPATIAL Workshop on Analytics for Local Events and News*, page 2. ACM, 2018.

[12] BBC. BBC news dataset. `http://mlg.ucd.ie/datasets/bbc.html/`, 2005.

[13] Eduardo Blanco, Nuria Castell, and Dan I Moldovan. Causal relation extraction. In *Lrec*, 2008.

[14] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[15] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[16] Jose Camacho-Collados and Mohammad Taher Pilehvar. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788, 2018.

[17] Xavier Carreras and Lluís Màrquez. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, 2004.

[18] Xavier Carreras, Lluís Màrquez, and Lluís Padró. Named entity extraction using adaboost, proceeding of the 6th conference on natural language learning. *August*, 31:1–4, 2002.

[19] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[20] Kunal Chakma and Amitava Das. A 5w1h based annotation scheme for semantic role labeling of english tweets. *Computación y Sistemas*, 22(3), 2018.

[21] Angel X Chang and Christopher D Manning. Sutime: A library for recognizing and normalizing time expressions. In *Lrec*, volume 2012, pages 3735–3740, 2012.

[22] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[23] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[24] Darrell Christian, Paula Froke, Sally Jacobsen, and David Minthorn. *The Associated Press stylebook and briefing on media law.* The Associated Press, 2014.

[25] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[26] Kevin Clark and Christopher D Manning. Improving coreference resolution by learning entity-level distributed representations. *arXiv preprint arXiv:1606.01323*, 2016.

[27] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[28] Ryan Compton, Craig Lee, Tsai-Ching Lu, Lalindra De Silva, and Michael Macy. Detecting future social unrest in unprocessed twitter data:emerging phenomena and big data. In *2013 IEEE International Conference on Intelligence and Security Informatics*, pages 56–60. IEEE, 2013.

[29] Alexis Conneau and Douwe Kiela. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*, 2018.

[30] Thomas M Cover, Peter E Hart, et al. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[31] Amitava Das, Sivaji Bandyaopadhyay, and Björn Gambäck. The 5w structure for sentiment summarization-visualization-tracking. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 540–555. Springer, 2012.

[32] Amitava Das and Björn Gambäck. Exploiting 5w annotations for opinion tracking. In *Proceedings of the fifth workshop on Exploiting semantic annotations in information retrieval*, pages 3–4. ACM, 2012.

[33] Amitava Das, Aniruddha Ghosh, and Sivaji Bandyopadhyay. Semantic role labeling for bengali using 5ws. In *Proceedings of the 6th International Conference on Natural Language Processing and Knowledge Engineering (NLPKE-2010)*, pages 1–8. IEEE, 2010.

[34] Franck Dernoncourt, Ji Young Lee, and Peter Szolovits. Neuroner: an easy-to-use program for named-entity recognition based on neural networks. *arXiv preprint arXiv:1705.05487*, 2017.

[35] Greg Durrett and Dan Klein. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982, 2013.

[36] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[37] G David Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[38] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[39] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[40] Tomohiro Fukuhara, Hiroshi Nakagawa, and Toyoaki Nishida. Understanding sentiment of people from news articles: Temporal sentiment analysis of social events. In *ICWSM*, 2007.

[41] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[42] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[43] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288, 2002.

[44] Roxana Girju. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering-Volume 12*, pages 76–83. Association for Computational Linguistics, 2003.

[45] Jose Gonzalez. Fuzzy Wuzzy: String matching. `https://github.com/seatgeek/fuzzywuzzy/`, 2018.

[46] Funda Güneş, Russ Wolfinger, and Pei-Yi Tan. Stacked ensemble models for improved prediction accuracy. In *SAS Conference Proceedings*, 2017.

[47] Felix Hamborg, Soeren Lachnit, Moritz Schubotz, Thomas Hepp, and Bela Gipp. Giveme5w: main event retrieval from news articles by extraction of the five journalistic w questions. In *International Conference on Information*, pages 356–366. Springer, 2018.

[48] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[49] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[51] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[52] George Hripcsak and Adam S Rothschild. Agreement, the f-measure, and reliability in information retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298, 2005.

[53] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[54] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014.

[55] IE. Indian Express. `https://indianexpress.com/`, 2019.

[56] Olaronke G Iroju and Janet O Olaleke. A systematic review of natural language processing in healthcare. *International Journal of Information Technology and Computer Science*, 8:44–50, 2015.

[57] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[58] Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier, 1997.

[59] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[60] Dan Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

[61] Christopher SG Khoo. *Automatic identification of causal relations in text and their use for improving precision in information retrieval*. PhD thesis, 1995.

[62] Sotiris B Kotsiantis. Bagging and boosting variants for handling classifications problems: a survey. *The Knowledge Engineering Review*, 29(1):78–100, 2014.

[63] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[64] Onur Kuru, Ozan Arkan Can, and Deniz Yuret. Charner: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 911–921, 2016.

[65] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[66] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[67] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[68] Kalev Leetaru and Philip A Schrodt. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, volume 2, pages 1–49. Citeseer, 2013.

[69] Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham. Svm based learning system for information extraction. In *International Workshop on Deterministic and Statistical Methods in Machine Learning*, pages 319–339. Springer, 2004.

[70] Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*, 2015.

[71] Xiaohua Liu, Kuan Li, Ming Zhou, and Zhongyang Xiong. Collective semantic role labeling for tweets with clustering. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[72] Edward Loper and Steven Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[73] Steven Loria. textblob documentation. 2018.

[74] Jing Lu and Vincent Ng. Event coreference resolution: A survey of two decades of research. In *IJCAI*, pages 5479–5486, 2018.

[75] Robert Malouf. Markov models for language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*, 2002.

[76] Christopher Manning, Richard Socher, Guillaume Genthial Fang, and Rohit Mundra. Cs224n: Natural language processing with deep learning1. 2017.

[77] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[78] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[79] Nancy McCracken, Necati Ercan Ozgencil, and Svetlana Symonenko. Combining techniques for event extraction in summary reports. In *AAAI 2006 Workshop Event Extraction and Synthesis*, pages 7–11, 2006.

[80] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[81] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[82] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[83] Sathappan Muthiah, Bert Huang, Jaime Arredondo, David Mares, Lise Getoor, Graham Katz, and Naren Ramakrishnan. Planned protest modeling in news and social media. In *Twenty-Seventh IAAI Conference*, 2015.

[84] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[85] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[86] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, 2007.

[87] Times of India. DMK members stage protests in front of ration shops across TN, detained. `https://timesofindia.indiatimes.com/city/chennai/dmk-members-stage-protests-in-front-of-ration-shops-across-tn-detained/articleshow/57615509.cms`, 2017. [Online; accessed 10-June-2019].

[88] Pamela E Oliver and Gregory M Maney. Political processes and local newspaper coverage of protest events: From selection bias to triadic interactions. *American Journal of Sociology*, 106(2):463–505, 2000.

[89] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning in natural language processing. *arXiv preprint arXiv:1807.10854*, 2018.

[90] Souneil Park, Seungwoo Kang, Sangyoung Chung, and Junehwa Song. Newscube: delivering multiple aspects of news to mitigate media bias. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 443–452. ACM, 2009.

[91] Kristen Parton, Kathleen R McKeown, Bob Coyne, Mona T Diab, Ralph Grishman, Dilek Hakkani-Tür, Mary Harper, Heng Ji, Wei Yun Ma, Adam Meyers, et al. Who, what, when, where, why?: comparing multiple approaches to the cross-lingual 5w task. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 423–431. Association for Computational Linguistics, 2009.

[92] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[93] Robi Polikar. Ensemble learning. In *Ensemble machine learning*, pages 1–34. Springer, 2012.

[94] Qiang Pu and Guo-Wei Yang. Short-text classification based on ica and lsa. In *International Symposium on Neural Networks*, pages 265–270. Springer, 2006.

[95] Lawrence R Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[96] Altaf Rahman and Vincent Ng. Narrowing the modeling gap: a cluster-ranking approach to coreference resolution. *Journal of Artificial Intelligence Research*, 40:469–521, 2011.

[97] Naren Ramakrishnan, Patrick Butler, Sathappan Muthiah, Nathan Self, Rupinder Khandpur, Parang Saraf, Wei Wang, Jose Cadena, Anil Vullikanti, Gizem Korkmaz, et al. 'beating the news' with embers: forecasting civil unrest using open source indicators. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1799–1808. ACM, 2014.

[98] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[99] Leonard Richardson. Beautiful soup documentation. *April*, 2007.

[100] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

[101] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[102] Yan Shao, Christian Hardmeier, and Joakim Nivre. Multilingual named entity recognition using hybrid neural networks. In *The Sixth Swedish Language Technology Conference (SLTC)*, 2016.

[103] Smriti Sharma, Rajesh Kumar, Pawan Bhadana, and Sumita Gupta. News event extraction using 5w1h approach & its analysis. *International Journal of Scientific & Engineering Research*, 4(5):2064–2068, 2013.

[104] Steve Stemler. An overview of content analysis. *Practical assessment, research & evaluation*, 7(17):137–146, 2001.

[105] Veselin Stoyanov and Jason Eisner. Easy-first coreference resolution. *Proceedings of COLING 2012*, pages 2519–2534, 2012.

[106] Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 2003.

[107] Leonard Talmy. *Toward a cognitive semantics*, volume 2. MIT press, 2000.

[108] Hristo Tanev, Jakub Piskorski, and Martin Atkinson. Real-time news event extraction for global crisis monitoring. In *International Conference on Application of Natural Language to Information Systems*, pages 207–218. Springer, 2008.

[109] TOI. Times of India. `https://timesofindia.indiatimes.com/`, 2019.

[110] Kari Torkkola. Linear discriminant analysis in document classification. In *IEEE ICDM Workshop on Text Mining*, pages 800–806. Citeseer, 2001.

[111] Manos Tsagkias, Maarten De Rijke, and Wouter Weerkamp. Linking online news and social media. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 565–574. ACM, 2011.

[112] Data Turks. Data turks. `https://dataturks.com/`, 2019.

[113] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[114] David L Waltz. An english language question answering system for a large relational database. *Communications of the ACM*, 21(7):526–539, 1978.

[115] Wei Wang. *Event Detection and Extraction from News Articles.* PhD thesis, Virginia Tech, 2018.

[116] Wei Wang, Dongyan Zhao, and Dong Wang. Chinese news event 5w1h elements extraction using semantic role labeling. In *2010 Third International Symposium on Information Processing*, pages 484–489. IEEE, 2010.

[117] Xiaofeng Wang, Matthew S Gerber, and Donald E Brown. Automatic crime prediction using events extracted from twitter posts. In *International conference on social computing, behavioral-cultural modeling, and prediction*, pages 231–238. Springer, 2012.

[118] Xing Wei and W Bruce Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185. ACM, 2006.

[119] Peter A White. Ideas about causation in philosophy and psychology. *Psychological bulletin*, 108(1):3, 1990.

[120] Sam Joshua Wiseman, Alexander Matthew Rush, Stuart Merrill Shieber, and Jason Weston. Learning anaphoricity and antecedent ranking features for coreference resolution. Association for Computational Linguistics, 2015.

[121] Phillip Wolff and Grace Song. Models of causation and the semantics of causal verbs. *Cognitive Psychology*, 47(3):276–332, 2003.

[122] Jiejun Xu, Tsai-Ching Lu, Ryan Compton, and David Allen. Civil unrest prediction: A tumblr-based exploration. In *International Conference on Social Com-*

puting, Behavioral-Cultural Modeling, and Prediction, pages 403–411. Springer, 2014.

[123] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[124] Vikas Yadav and Steven Bethard. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158, 2018.

[125] Vikas Yadav, Rebecca Sharp, and Steven Bethard. Deep affix features improve neural named entity recognizers. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 167–172, 2018.

[126] Sibel Yaman, Dilek Hakkani-Tür, Gokhan Tur, Ralph Grishman, Mary Harper, Kathleen R McKeown, Adam Meyers, and Kartavya Sharma. Classification-based strategies for combining multiple 5-w question answering systems. In *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[127] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[128] Zhibin Zhao, Jiahong Sun, Zhenyu Mao, Shi Feng, and Yubin Bao. Determining the topic hashtags for chinese microblogs based on 5w model. In *Interna-*

tional Conference on Big Data Computing and Communications, pages 55–67. Springer, 2016.