Discover New Actors In Politics: A Framework To Recommend Political Actors With Role In Real-time

Mohiuddin Solaimani[1], Sayeed Salam[1], Latifur Khan[1],
Patrick T. Brandt[2] and Vito D'Orazio[2]
[1]Dept. of CS, [2]School of Economic, Political, and Policy Science
The University of Texas at Dallas
(mxs121731,sxs149331,lkhan,pbrandt,dorazio)@utdallas.edu

ABSTRACT

Extracting a structured representation of events (political, social etc.) has become an interesting domain in the computational and social sciences. Traditional approach is to use dictionary-based pattern lookups to identify actors and actions involved in potential events represented in who-did-what-to-whom format. A key complication of this approach is updating the dictionaries with new actors (e.g., when a new president takes office). Currently, the dictionaries are curated by humans, updated infrequently, and at a high cost. This means that tools dependent on the dictionaries (e.g., PETRARCH) overlook events because they are missing dictionary entries. In this paper, we address how to extend the dictionaries used to identify new political actors using Automatic Content Extraction (ACE). We extract event with PropBank (a semantic role labeling tool). We propose a framework to recommend new political actors from online political news articles periodically. We develop a frequency-based actor ranking algorithm using partial string matching based (e.g., Levenshtein/Edit distance, MinHash, etc.) actor grouping for dynamic new actor recommendations over multiple time windows. Moreover, we suggest the associated evolving role of recommended actors from the role of co-occurred political actors in the existing CAMEO actor dictionary. Furthermore, we consider PETRARCH and BBN ACCENT event coders for actor recommendation, and a graph-based actor role recommendation using weighted label propagation as baselines and compare them with our framework. Experiment results show our approaches outperform them significantly.

CCS CONCEPTS

•Information systems → Data extraction and integration; *Data stream mining;* **•Applied computing → Document management and text processing;**

KEYWORDS

PropBank, PETRARCH; SPARK; CAMEO; Actor Ranking

1 INTRODUCTION

Political event data [8] are encoded from news reports. This can be accomplished manually by humans or via machine coding. Events are categorized based on a set of dictionaries for the actions and actors. The typical format to create these data is to determine "who-did/said-what to whom" [1]. While the set of actions or verbs is finite and can be matched to a fixed ontology, the set of source or target nouns is quite large in political terms. To transform news reports into useful data to study international relations and civil conflict, identifying these political actors along with their roles is important.

Currently, adding new actors and their roles to the dictionary are done by Humans. Automated coders (i.e. PETRARCH [1]) use those dictionaries to identify events. However, if a source or target actor has *not* been entered by a human in the relevant actor dictionary, PETRARCH will ignore the event related to the new actor. So we lose valuable information. To facilitate PETRARCH, we need automatic content extraction techniques, so that we can find possible new political actors from the events that PETRARCH ignores. That motivates us to design a framework for recommending political actors in real-time.

Automatic Content Extraction (ACE) programs [12] infer entities, relations, and events from human language data. An event consists of ACE relation which has a number of participants (ACE entities) and each participant has a semantic role that it plays in the event (agent, object, source, target). Machine learning-based semantic role labeling [14] technique consists of the detection of the semantic arguments associated with the predicate or verb of a sentence and their classification into their specific roles. Example of tools implementing such technique includes FrameNet [7], PropBank [24], VerbNet [16]. Among them, PropBank Corpus provides structured predicate-argument annotation for the entire Penn Treebank. Each verb in the treebank is annotated by a single instance in PropBank and acts like an event containing information about the source, target, and location of the verb.

Designing the system poses several challenges. First, actors may come with multiple alias names, e.g., 'Barack Hussein Obama', 'Barack Obama', 'President Obama', etc. Currently, humans include a dictionary entry for each of these alias in CAMEO actor dictionary. Second, the role of an actor changes over time. For example, 'Shimon Peres' served multiple political roles in Israel during his lifetime. Finally, processing a large volume of news articles across the world demands scalable, distributed computing to detect these changes.

Considering these challenges, we develop a real-time, distributed recommendation framework for the identification of potential actors and their associated roles. We develop this to reduce human efforts in the dictionary update and development process. We gather political news articles from all-around-the-world, pre-process them with Stanford CoreNLP and PropBank to extract the semantic information needed about the actors. Then we apply our unsupervised ranking algorithm for recommending probable new actors with role. Our actor recommendation uses PropBank for event coding, and CAMEO action dictionary to classify political events. We compare it with PETRARCH, and a proprietary event coder BBN ACCENT™

[9] and show the performance. Moreover, we implement a graph-based actor role recommendation technique using weighted Label propagation [19] as baseline and compare it with our role detection method. The paper highlights the following contributions.

First, we have proposed a novel time window-based, unsupervised new actor recommendation technique with possible roles. In particular, we have designed a frequency-based actor ranking algorithm with alias actor grouping from news articles.

Second, we have developed a distributed real-time framework using Apache Spark Streaming [36] to address the scalability issue of event coding over political news articles.

Finally, we have compared our proposed approaches with state of the art (e.g., PETRARCH, BBN ACCENT event coding) and shown the effectiveness of our work.

The rest of the paper is organized as follows: Section 2 gives the background on the various open source tools and concepts used in this paper. Section 3 explains the overall framework. Section 4 describes our new actor with role recommendation technique in details. Section 5 shows the experimental results for new actor detection. Section 6 covers the related works. Section 7 states the conclusion and future works followed by a bibliography.

2 BACKGROUND

Stanford CoreNLP [31] is an useful tool for annotating text with part-of-speech (POS) taggers, named entity recognition (NER), parser, co-reference resolution system, sentiment analysis, and bootstrapped pattern learning tools.

CAMEO (Conflict and Mediation Event Observations) is an event coding framework to record political events (both material and verbal interactions) in a structured who-did-what-to-whom format. It was originally intended to the study of interstate conflict mediation [28]. Over time, it has developed as a "next generation" coding scheme for automated coding and the detailed coding of sub-state actors. There are dictionaries for actors and actions (verbs) which form the knowledge-base for the framework. Tools using this framework first consider the structure of the sentence to identify possible actors and actions, then they look for their existence in the corresponding dictionaries. Once found, they can use an appropriate actor and action coding to represent the event. Each action in CAMEO is represented with a designated numeric code. Each actor is represented with corresponding actor codes (i.e., GOV, MIL, USA, etc.). These are associated with the actors based on a timeline since the role of an individual can change when they achieve a new position in government or other agencies.

PropBank [24] is a corpus which annotates text semantically. It adds a semantic layer to Penn Treebank [33] Attempts to capture accurate predicate-argument structure by annotating predicates and the semantic roles of their arguments.

Apache Spark [36] is an open-source distributed framework for data analytics. It avoids the I/O bottleneck of the conventional two-stage MapReduce programs by providing in-memory cluster computing. Also, it supports both batch processing and streaming data.

MongoDB [21] an open source NOSQL document database. MongoDB stores data in the form of a BSON (Binary JSON-like) document. A set of documents forms a collection.

Word2vec [20] is a two-layer neural net text processing tool. It takes a text corpus as input and its output is a set of vectors: feature for words in that corpus. Word2vec groups the vectors of similar words together in vector space and detects similarities mathematically. Word2vec creates vectors that are distributed numerical representations of words as features.

3 FRAMEWORK

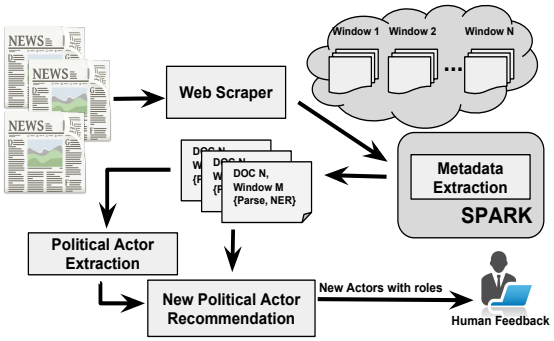


Figure 1: Framework for real-time new political actor recommendation

Figure 1 shows our new political actor recommendation framework. It collects political news stories periodically through web scrapers. Later, it uses CoreNLP and PropBank to extract metadata from the stories like parse trees, tokens, lemmas, NER, semantic events (by PropBank), etc. and stores them in MongoDB. We use Apache Spark streaming with Kafka [17] to collect all the scrapped data, periodically. We use CoreNLP an PropBank annotation inside a Spark worker node to scale up the process. After that, it classifies political events only using CAMEO action dictionary from the metadata. It fetches possible new actors from source and target of the events and NER. We implement a window-based, unsupervised frequency-based actor ranking technique to recommend potential new actors with their related roles using event codes and associated metadata. Finally, human experts validate the recommended actors with roles and update the dictionary. We describe our framework in details in following subsections.

3.1 Web Scrapper

We use a web scraper [23] to collect news articles and extract the main content of the news. The web scraper has been integrated into the rest of the framework and runs periodically (e.g., every 2 hours). It collects news stories from about 400 RSS (Rich Site Summary) Feeds[4] (on average, each contains 15-20 links to news articles). The news articles are shipped through Apache Kafka to an Apache Spark-based data processing module[30].

3.2 Metadata extraction

Inside our data processing unit, CoreNLP parses all the news documents and extracts metadata like Parts-Of-Speech (POS) tagging, Parse Tree, Named Entity Recognition (NER) etc. and stores them into MongoDB. We use NER for our framework.

Named-entity recognition (NER) [22] locates and classifies named entities in text into predefined categories such as the names

of persons, organizations, locations, etc. In our framework, we use persons and organizations to detect new actors. We use NER given by Stanford CoreNLP because it has higher accuracy [5, 26] than other similar tools like Alchemy [15], OpenCalais [34], OpenNLP [32], etc.

Event coding. PropBank generates event [6] with 'predicate-argument' structures for each sentence for a given text. In general, it has the following arguments correspond to the following semantic roles of agent/source (ARG0) and target (ARG1, ARG2, etc.). It has some extra arguments (annotation of modifier) that describe time (AM-TMP), location (AM-LOC), etc.

Table 1 shows the generated events for the following example shows such an event.

"Obama vowed again on Sunday to help France hunt down the perpetrators of the attacks." - The Washington Post 11/15/2015

Table 1: PropBank annotation example

Verb	ARG0	ARG1	AM-TMP
vowed	Obama	to help France hunt down the perpetrators of the attacks	again, on Sunday
help	Obama	France hunt down the perpetrators of the attacks	
hunt	France	down the perpetrators of the attacks	

3.3 Political Actor Extraction

PropBank provides events with arguments. Each of them consists of a verb/action which is looked up from CAMEO dictionary to categorize political event. Each political events contains arguments like ARG0 as a source, ARG1, and ARG2 as a target. We parse these arguments and filter name entities (Person, Organization) with NER that are missing in CAMEO actor dictionary. These actors are possible new actors. We detect an actor with alias names and group them together.

3.4 New Political Actor Recommendation

We introduce a window based actor ranking where we select top N actors for each window. We recommend an actor as a new actor if he/she appears in most of the windows. New actors role can be inferred from their related existing actors' roles. We describe them in details in section 4.

3.5 Human Feedback

We provide a graphical user interface and dashboard to the end users/human expert. It shows all the new recommended actors with their role. Users can easily view them and update actor dictionary (e.g., CAMEO) accordingly.

4 RECOMMENDING A NEW ACTOR AND ROLE

Figure 2 describes the technical details of our real-time actor recommendation technique. We use NER to filter out all possibles actors from a news article. After that, we calculate a score or ranking for each actor across all the documents and select the top N with

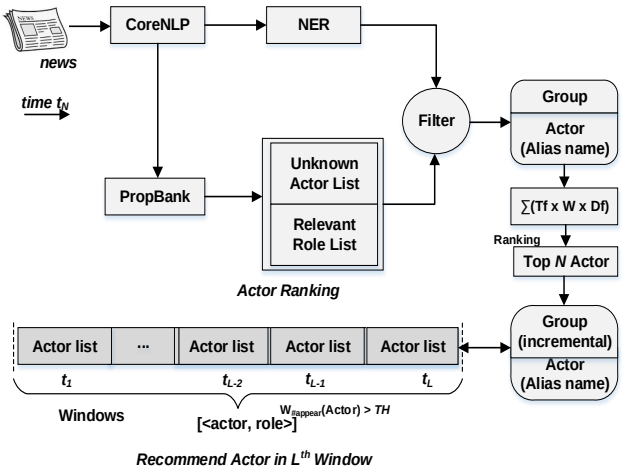


Figure 2: Technical details.

their roles. Here we have maintained a buffered time window W of length L. Each window contains the top N actors, which are merged with the previous windows actors list and updated the rank statistics. After L windows, we recommend new actors with their role, if their occurrences in the L_{th} buffered window exceed a certain threshold TH. We describe in details about this in forthcoming subsections starting with the following example.

Scenario - Donald Trump is the new President of United States. Current CAMEO actor dictionary does not have any entry of him. So, current event coders (e.g., PETRARCH) will not generate any event code from news articles related to Donald Trump unless human updates dictionary manually. On the other hand, our real-time framework periodically captures political news and from them, it identifies new probable political actors. As many news highlight Donald Trump, he has a high chance of being an actor and we identify and recommend him as a political actor. Moreover, we suggest his role from his surrounding existing co-occurred political actors.

4.1 New Actor Discovery

We use a frequency-based ranking algorithm to recommend the top political actors and their roles. Each news article/document contains raw text related to a political event. CoreNLP parses this raw text and outputs NER for each sentence. PropBank generates event codes with arguments for each sentence using practnlptool [2]. These event codes' arguments (ARG0, ARG1, ARG2, etc.) represent sources/targets of relations/events. From these arguments, we filter out name entities list using NER. Sometimes, this list contains existing actors and their roles can be looked up from CAMEO dictionary. For example, *President Obama* has role *USAGOV*. NER gives us all actors (i.e Person, Organization) in a sentence. A news article contains multiple sentences. We filter potential new actors by removing existing actors from NER actor list. Sometimes, an actor may come in different name/partial name across the article. For example, *Barack Hussein Obama* may come as *Obama* or *Barack*

Table 2: Symbol for algorithm

D	document set	d	document
m	meta-data	sim_{th}	similarity threshold
E	event code set	e	event code
A	actor set	a	actor
R	role set	r	role
tf	term freuency	df	document frequency
N	top N actor	L	number of window
k	merged actor as key	TH	Threshold in window

Obama in many sentences. As a result, we need to group these actors with all possible alias names and make a single actor name (e.g., *Barack Hussein Obama*). We use several similarity measures like Levenshtein Distance (Edit distance) [18], MinHash [10] to group alias actors. All of these methods need a similarity threshold sim_{th}. After grouping, we calculate the term-frequency of each actor inside the news article. For an alias actor, we take the maximum count as its term-frequency because we group actor alias names. We assume that a common word from all alias names will be found throughout the document. For example, the term-frequency of *Barack Hussein Obama* will be the maximum frequency of *Barack Hussein Obama, Obama, and Barack Obama* as Obama is the common word between all names. Similarly, we group aliases for actors across multiple documents. Finally, for each actor a, we calculate a document frequency df and use the following equations to generate a score:

We use term frequency $tf(a,d) = count_{a \in d}(a)$ to show how frequent an actor a in a document d. Equation 1 shows the rank calculation of an actor a, where $df(a,D) = \frac{|d \in D : a \in d|}{|D|}$ is document frequency and it shows how frequent an actor a comes across all news articles/document in the set D.

$$rank(a) = \sum_{d \in D} tf(a,d) \times df(a,D) \tag{1}$$

4.2 Actor role discovery

Our framework extracts political events for sentences in a news article. We collect all events which contain new and existing political actors. CAMEO actor dictionary contains existing actors with their roles. Here, we assume that new actors' political roles will be related to their most frequent co-occurred existing political actors. So, we collect the roles and assign them to new actors as possible roles. We keep a map of each role with its occurrence. We update actors' maps with a role for all news articles. When an actor appears in two news articles, we include all roles in both articles and increase common roles' occurrence. While we have all ranked actors, we also have their role maps.

Algorithm 1 shows the new actor discovery algorithm with ranking. It takes the news document set D as input and gives actor map with rank M_{rank} as output. M_{rank} contains each actor as key with its ranking and role map M_{role}. It takes each document $d \in D$ and extracts new actor with roles R_d (lines 4-8). After that, it groups all aliases for actors inside a document with counting term-frequency tf (lines 10-16). Next, it groups actors' aliases across multiple documents and updates the temporal actor map M which contains an actor as a key and a list of the tuple $< tf, d, R_d >$ as a value (lines 17-24). After that, for each actor in M, it calculates the document frequency d, and $rank$ score using equation 1 (lines 28-29). It then creates a role map M_{role} for each actor which contains all the possible roles with their total occurrences across the document set D (lines 31-35). Finally, we insert the actor as key and a tuple of $rank$, and role map M_{role} as value into M_{rank}. So, M_{rank} contains all possible new actors with their $rank$, and possible roles.

Algorithm 1 New actor discovery

```
1:  procedure ACTORDISCOVERY(DocumentSet D)
2:      Actor Map M ← {}
3:      for each d ∈ D do
4:          m ← CoreNLP(d)
5:          E_d, A_current ← PropBank(d)
6:          A_all ← m.NER()
7:          A_d ← A_all − A_current
8:          R_d ← E.Roles()
9:          for each a ∈ A_d do
10:             tf ← count_{a∈d}(a)                    ▷ calculate tf
11:             for each a_i ∈ A_d − a do
12:                 if Match(a, a_i) > sim_th then
13:                     a ← Merge(A_d, a, a_i)
14:                     tf ← max(tf, count_{a_i∈d}(a_i))
15:                 end if
16:             end for
17:             k ← arg max_k {Match(M(k), a) > sim_th}
18:             if k ≠ empty then
19:                 k ← Merge(M, k, a_i)
20:                 M.insert(k, [(tf, d, R_d)])
21:             else
22:                 M.insert(a, [(tf, d, R_d)])
23:             end if
24:         end for
25:     end for
26:     M_rank ← {}
27:     for each k ∈ M.keys() do
28:         df ← |k ∈ d : d ∈ D|/|D|                   ▷ calculate df
29:         rank ← Σ_{tf∈M(k)}(tf) × df
30:         M_role ← {}
31:         for each R_d ∈ M(k) do
32:             for each r ∈ R_d do
33:                 M_role(r) ← M_role(r) + 1
34:             end for
35:         end for
36:         M_rank.insert(k, (rank, M_role))
37:     end for
38: end procedure
```

4.3 Recommending new actors in real-time

Our framework gets political news articles periodically and possesses a buffered time window W of length L. At each time window, it scans all the news articles and gets the possible actor list with rankings and possible roles. It is well known that new political actors will always be highlighted in media and people will always talk about them and in print or online media. Here we assume that, if an actor comes in the top N ranking in multiple time windows, he or she has a high probability of being a new political actor.

Algorithm 2 Real-time new actor recommendation

1: **procedure** ACTORINREALTIME(TH_{min}, TH_{max}, N, L)
2: *New Actor Map* $M_A \leftarrow \{\}$
3: *New Actor Role Map* $M_R \leftarrow \{\}$
4: **for** $t \leftarrow t_1$ *to* t_L **do**
5: $D \leftarrow getDocuments(t)$
6: $M_{rank} \leftarrow ACTORDISCOVERY(D)$
7: $M_{topN} \leftarrow M_{rank}.top(N)$
8: **for each** $< a, (rank, M_{role}) > \in M_{topN}$ **do**
9: $k \leftarrow \arg\max_k \{Match(M_A(k), a) > sim_{th}\}$
10: **if** $k \neq empty$ **then**
11: $k \leftarrow Merge(M, k, a)$
12: $M_A(k) \leftarrow M_A(k) + 1$
13: **else**
14: $M_A(a) \leftarrow 1$
15: **end if**
16: **for each** $r \in M_{role}$ **do**
17: $M_R(k) \leftarrow M_R(k) + 1$
18: **end for**
19: **end for**
20: **end for**
21: **for each** $a \in M_A$ **do**
22: **if** $M_A(a) \geq TH_{max}$ **then**
23: $M_A(a) \leftarrow$ "*new actor*"
24: $M_R(a) \leftarrow$ "*new actor's role*"
25: **end if**
26: **if** $M_A(a) \leq TH_{min}$ **then**
27: $M_A(a) \leftarrow$ "*discard actor*"
28: $M_A.remove(a)$
29: $M_R.remove(a)$
30: **end if**
31: **end for**
32: **end procedure**

Algorithm 2 describes in details about the recommendation procedure. For each time window, we receive the actor list with ranking M_{rank}. We take the top N actors from the list (lines 5-7). We update a new actor Map M_A and its role map M_R. These are incrementally updated during each time window. For each actor in M_{rank}, we find the list of co-occurred actors in M_A. If there is no actor in M_A, we insert it in M_A with occurrence 1 and its role in M_R. If we find the closest match, then we merge the actor name if required. Later, we increment its occurrence in M_A. In the same way, we update the role of that actor in M_R (lines 8-20). As we increase the actors in M_A and M_R, their size will always be increased. So, we introduce a (min, max) threshold TH_{min} *and* TH_{min}. After the L^{th} time window, if the total occurrence of an actor is greater that TH_{max}, we consider his/her as a new actor and recommend top N roles from M_R also. On the other hand, it total occurrence is below TH_{min}, we discard the actor from M_A and M_R (lines 21-31). After L^{th} time window, we will recommend new actors with possible related roles. Human experts will verify this and update CAMEO dictionary accordingly.

5 EXPERIMENTS

5.1 Setup and Dataset

To evaluate our framework, we take 9-time windows at 2 hours interval. In each time window, we scrap newly published online news articles listed in 400 RSS feeds around the world. We process these articles/documents with CoreNLP, practnlptools running in Apache Spark. Finally, the processed articles are used for recommendation-related tasks.

5.2 Threshold Estimation

We are using similarity measure for grouping actors with alias names. This uses partial string matching. In the experiments, we use both MinHash and Edit distance and show their performance. Both approaches give a numeric value between 0 and 1 indicating the similarity. For our purpose, we need to set a threshold value for considering whether two strings represent the same actor using these methods. When the similarity value between two actor names becomes greater than the threshold, we regard them as the same entity. To estimate the threshold, we consider similarity values for actors already in the CAMEO actor dictionary. As an example, consider the following snippet from the actor dictionary for Kofi Annan and Boutros Boutros Ghali:

```
....
KOFI_ANNAN_
+SECRETARY-GENERAL_KOFI_ANNAN
+UNITED_NATIONS_SECRETARY_GENERAL_KOFI_ANNAN
.....
BOUTROS_BOUTROS_GHALI_
+BOUTROS_BOUTROS-GHALI_
+SECRETARY-GENERAL_BOUTROS-GHALI_
....
```

Both of them are followed by multiple aliases (lines starting with a '+'). We calculate the similarity between these aliases and the aliases of different actors. In the first case, we observe higher values (close to 1). For the later case, we found smaller values. Then we set a value that will minimize the number of false positives (i.e., reporting high similarity when they are not same actor). By this empirical study, we estimate thresholds for Edit distance and min-hash based methods to be 0.75 and 0.4 respectively.

5.3 Baseline methods

We consider the following methods as baselines.

PETRARCH [1] takes fully-parsed text summaries in Penn Tree format [33] from Stanford CoreNLP and generates events in a 'who-did-what-to-whom' format. It outputs source, target and a CAMEO code [11] for the political event along with other information like the date of an event, news source, etc. We used it as a baseline event coder and apply a frequency based approach to recommend new actors. Finally, we compare it with our approach in terms of both precision and recall.

BBN ACCENT^{TM} [9] event coder automatically extracts event data from news reports from around the world. It is based on *BBN SERIF*^{TM}, a natural language analysis engine that extracts structured information (e.g. entities, relationships, and events) from a text. Its events are coded according to the Conflict and Mediation

Event Observations (CAMEO) ontology. It is a proprietary political event coding tool developed by Raytheon BBN Technologies Corp [3]. BBN encodes political event which has actor, agent, and action (CAMEO event type) but it does not generate the role of an actor. As BBN is proprietary and its used dictionaries are encrypted, so our comparison is based on new actor recommendation with human expert validation, i.e., precision comparison. This means that our framework will recommend new actors using Propbank, PETRARCH, and BBN separately and a human expert will validate how many of them are really new actors for each of the event coding tools.

Graph-based Role detection technique We introduce a graph-based technique to suggest roles for recommended actors. Roles of a new actor will be influenced by existing actors with whom he/she has mostly interacted. Therefore, we use weighted label propagation technique[19] to infer possible roles from existing related political actors.

We formulate a graph $G = (V, E)$ that implies the interaction between actors. Here, V is the set of actors (contains both existing and recommended actors). For two actors, u and v, $(u, v) \in E$ represents those actors are mentioned in the same news article. We also assign a weight function $w(u, v)$ that implies how frequent u and v are co-occurred in the news articles.

$$w(u, v) = \text{co-occurrence count of } u \text{ and } v$$

After that, we assign labels to the nodes which are the roles found in the dictionary for existing actors. For recommended actors, we simply put an empty label. We use $label(x)$ to denote the label for actor x.

$$label(x) = \begin{cases} \text{roles from dictionary} & \text{if } x \text{ is an existing actor} \\ empty & \text{otherwise} \end{cases}$$

Now we run our iterative label propagation algorithm, where we assign weight to each possible role for a recommended user. Roles are selected from neighboring actors. Weight is assigned for each role based on a weight associated with the edge of interaction graph, G. We repeat the process N times or until the label assignments become stable. The weighting function for roles works as follows,

$$role - weight(u, role) = \sum_{v \in Neighbors(u)} T(v, role) \times w(u, v)$$

where $T(v, r)$ is used as indicator variable that simply returns 1 if v has 'r' in it's list of roles, 0 otherwise. In figure 3, we will determine

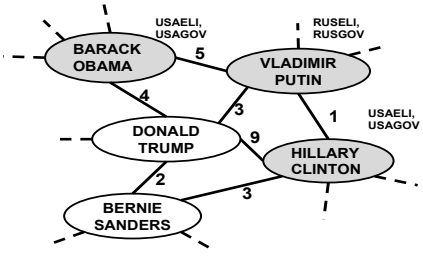


Figure 3: Example Scenario for Graph Based Role Recommendation

the role for DONALD TRUMP. He is associated with both existing actors (gray nodes) and new political actors. All the existing actors have their roles associated with their nodes. For example, BARACK OBAMA has USAELI and USAGOV as his roles. Now, we use our role-weight method to calculate the role. For example,

$$\text{role-weight(DONALD TRUMP, USAGOV)}$$
$$= 4 \times 1 + 9 \times 1 + 3 \times 0 + 2 \times 0 = 12$$

After calculating weights for all roles from neighbors, we find USAGOV, USAELI, RUSGOV as the top 3 possible roles for DONALD TRUMP.

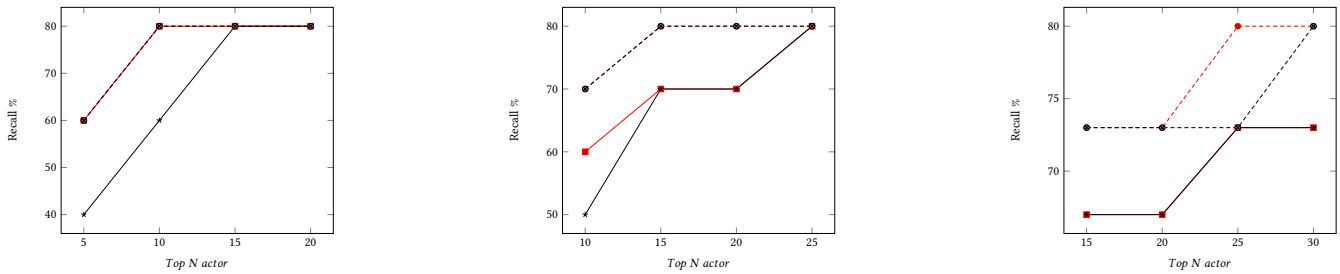
5.4 Experiment 1: Performance Evaluation

This experiment evaluates the framework's correctness. To do this, we eliminate some well-known actors from existing CAMEO actor dictionary and try to retrieve them using our framework over time. We start with removing 5, 10 and 15 actors. The deleted actors are considered for a recommendation (as if they are newly discovered). We calculate how many of them are recommended by our algorithm. This gives us the recall. We can not compute precision because all the recommended actors are political actors.

From figure 4, we see that if we increase the top N actors (i.e., how many actors are recommended at each time window), it will increase the possibility of retrieval of a deleted actor. Because there are actors who are not highly focused by media but have a continuous impact in his/her locality.

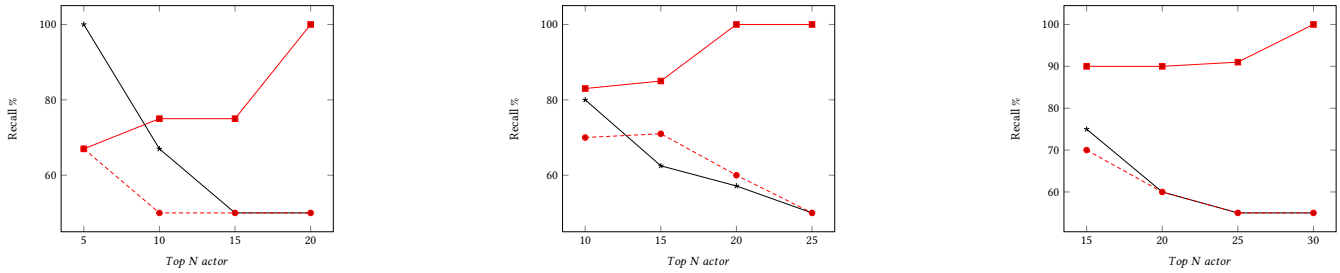
We also see that our framework which has PropBank based event coder (dotted line) has higher recall than PETRARCH based event coder (solid line) for recommending new actors. This is expected because PETRARCH misses many verbs as event while coding. For example, table 1 in section 3.2 shows three separate events using PropBank but PETRARCH will generate only one event 'FRA(Source), USAGOV(Target),Cooperate(Verb)'. As a result, our framework will generate more new actors that PETRARCH. Moreover, Edit distance based (red) actor alias grouping has higher recall than MinHash based (black) actor alias grouping for both approaches because Edit distance considers each character where MinHash consider each word while comparing alias names of an actor.

In the following experiment, we suggest the possible roles for identified actors in the previous experiment. As a reference, we have their roles (R1) listed in the CAMEO dictionary. Now using our frequency-based role recommendation algorithm we predict appropriate roles (R2) for them. We calculate the intersection of the two sets, R1 and R2. If the output is non-empty set then we consider it as a success. For the purpose of the experiment, we keep the size of R2 as 5. The roles are selected based on frequency. Top 5 frequent roles are presented as possible actor roles. Again we vary the number of deleted actors. We also vary the top-N. For the experiment, we reported the ratio of a number of success and number of actors retrieved. Figure 5 shows the results. We see that Edit distance has a slight advantage while actor recommendation. We use word set in MinHash calculation and it uses Jaccard similarity. So, for close matched actor aliases like 'Donald Trump' and 'Donald J Trump', Edit distance gives higher similarity than MinHash. For similar reason, we see recall variation in figure 5 for role recommendation. Roles of an actor is a short text. Edit distance performs better when top N increases, whereas MinHash and Exact



(a) Deleted actor = 5 (b) Deleted actor = 10 (c) Deleted actor = 15

Figure 4: Performance for Actor recommendation. Recall: Edit distance (PropBank), MinHash (PropBank), Edit distance (PETRARCH), MinHash (PETRARCH)



(a) Deleted actor = 5 (b) Deleted actor = 10 (c) Deleted actor = 15

Figure 5: Performance for role recommendation. Recall: Edit distance, MinHash, Exact match

matching have similar level of performance (worse than Edit distance). Because for Edit distance, we consider partial matching. For example, USAMILGOV and USAMIL are similar using Edit distance method, but hugely different when MinHash (Jaccard similarity) is used. They are certainly different for Exact Matching.

Discovering roles with Word2vec. We use Word2vec to find the related roles, which requires a training model. We take possible new actors with related existing actors from news articles. We collect them across 1 *to* L^{th} (algorithm 2) time windows to train the model. For each new actor, we find the top N similar existing actors (in the CAMEO dictionary) using Word2vec's *similal_word* method and recommend their roles. Unfortunately, we get less than 5% recall. This is obvious because Word2vec does not apply alias actor grouping. As a result, it fetches less similar actors than our frequency-based approach.

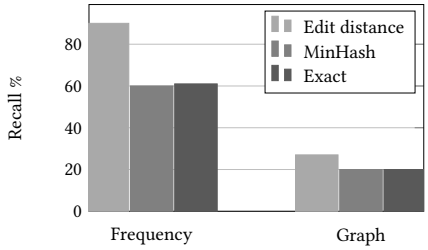


Figure 6: Comparison of actor role recommendation with baseline: (N = 15, deleted actors = 15)

Comparison of Role Recommendation Techniques. In this experiment, we first delete some well-known actors from the existing CAMEO dictionary. Then we try to retrieve their roles by frequency based and **graph based** role detection techniques. We fix the values of numbers of recommended actors per window and number of deleted actors to 15. We do not vary those parameters for graph-based approach because it caused the resulting graph to be very sparse. That in-turn makes the role inference hard with label propagation technique. For example, if we use $N = 5$ and number of deleted actors=15, we found only 40 actors with 56 edges between them in the interaction graph. That is only 7% of the edges possible in a complete graph with the same number of nodes. So, we consider all the actors suggested by the actor recommendation algorithm and their interactions to formulate the graph. Figure 6 shows the statistics for the experiment. Here, frequency based approach outperforms graph-based approach for all the similarity measurement techniques because it infers new actor's role from existing actors' roles. But in the case of graph-based approach, it considers roles from neighbors who can be either existing or new actors. In that case, the error with one role assignment can propagate to others. The output contains 91 existing actors with 694 possible new actors. That means label propagation algorithm have less number of labels to propagate and the edges between new actors will be higher because they have a larger possibility to co-occur in a document.

5.5 Experiment 2: Recommendation of New Actors with Roles

Here, we list the possible newly recommended political actors based on a threshold. The threshold is how many times an actor appeared in the time windows. We set the number of time window L = 9 and (max, min) frequency threshold (TH_{max}, TH_{min}) = (5, 2) in algorithm 2. So if any actor appears in 5 (more than 50% window length), he/she will be suggested as a potential new political actor. In the case of roles, we list the most probable roles from their co-occurred existing political actors. Table 3 shows the list of recommended actors after all windows. We use both MinHash and Edit distance based string similarity and list the output side-by-side. We found that both the approach detects similar roles for identical actors. But the recommended user list varies.

Compare with PETRARCH and BBN ACCENT. We use PETRARCH and BBN ACCENT as event coder in our framework and run experiments to recommend new actors. Human expert validates recommended new actors and we calculate precision based on that. We run the same experiment using our framework and calculate precision. We vary the number of top N recommended actors. Figure 7 shows the comparison. It shows that our framework (PropBank based event coder) has higher precision (dotted red) than PETRARCH (solid red) and BBN ACCENT (solid black) coding. If BBN ACCENT fails to find any actor in the dictionary, it generates an empty event. On the other hand, PETRARCH separates political and nonpolitical events with actors but misses many verbs as event compares to PropBank based coding.

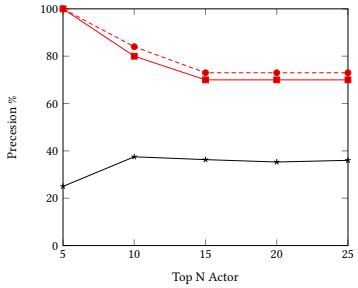


Figure 7: Baseline comparison with actor detection: PETRARCH coding —■—, BBN ACCENT —⋆— coding, and PropBank - -●- - coding

5.6 Experiment 3: Scalability Test

We used our own Spark-based cluster with streaming data. Table 4 shows the configuration of the cluster.

We are using CoreNLP which is computationally expensive [30]. Each CoreNLP instance runs inside a Spark worker node and extracts metadata from the news article. Figure 8 shows the scalability of average document processing time of 5776 news articles using Spark. Spark fetches the news articles at 10 minutes intervals, whereas the scrapper sends news at 2 hours interval. For this test, we take 5776 news articles in a burst to observe actual performance. Normally, the scraper fetches 5000-6000 online news article at the beginning of a run. Later, it scrapes only the updated news articles which are fewer than initial. Spark internally creates micro-batches

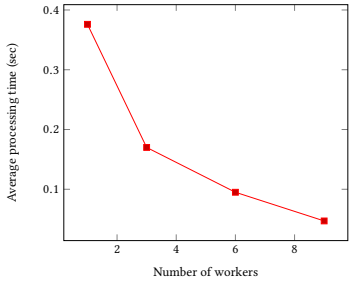


Figure 8: Processing time for burst input of 5776 documents

to process these 5776 articles. So, we take average processing time per article. If we have only one worker, then its average processing time is almost 0.376 sec which is reduced to 0.047 when the total number of worker reaches to 10 (maximum load). It scales up almost linearly.

6 RELATED WORK

Political event data analysis has been developed over multiple decades for international relations and security studies. Saraf et al. [27] showed a recommendation model to determine if an article reports a civil unrest event. Schrodt and Van Brackle [29] show a complete picture of generating events from raw news texts. While each of these works focuses on political event data generation and analysis, none incorporate dynamic dictionary building. Beieler et al. [8] point out that this is a major challenge to developing and extending event data. Yet, as discussed, actor dictionaries are crucial to construct political event data. Currently, researchers manually update the CAMEO actor dictionaries when new entries are needed. This is a labor-intensive process that does not scale.

Role recommendation based on surrounding keywords in the document can be considered. For example, 'President' keyword occurring in front of Barack Obama may help infer his role. But we may not get enough keywords like above to infer roles. Also often they do not convey the complete role for an actor. Our above example shows similar scenario. With the keyword 'President', we can imply Barack Obama as a government employee. But we cannot infer his country. Moreover, CAMEO uses short-form encoded roles (e.g., USAGOV) which require a static mapping if we consider the above procedure.

Automatic Content Extraction (ACE) based event extraction systems use pattern or machine learning. Ritter et al. [25] proposed TWICAL - an open domain event extraction and categorization system for Twitter. Georgescu et al. [13] showed event extraction using Wikipedia. Weninger et al. [35] shows future Web content extraction algorithms. All the above methods use classifiers to identify event type which performs well in a generic domain but not in a restricted political domain. Therefore, we use a mixed model of CAMEO and PropBank to extract political events.

Extracting events from raw news articles in real-time demand a scalable distributed streaming framework. We choose Spark Streaming because it is matured and widely used in industry.

7 CONCLUSION AND FUTURE WORK

In this paper, we address the problem of detecting and recommending new political actors and their roles in real-time. We propose a

Table 3: List of recommended actors with their roles

Edit distance		MinHash	
Actor	**Top 3 roles**	**Actor**	**Top 3 roles**
DONALD TRUMP	USA, USAELI, LEG	SEDIQ SEDIQQI	AFG, PPL, UAF
SEDIQ SEDIQQI	UAF, AFG, PPL	DONALD TRUMP	USA, USAELI, LEG
AMIR SHEIK SABAH AL-AHMAD AL-JABER AL-SABAH	SAUMED, SAUGOV, KWTMEDGOV	AMIR SHEIK SABAH AL-AHMAD AL-JABER AL-SABAH	SAUMED, SAUMEDGOV, KWTMEDGOV
WILLIAM HUTCHINSON	USA, MIL, REB	BUKOLA SARAKI	USAGOVLEG, GOV, NGAELIGOV
		CLEMENT MOUAMBA	COG, PPL, GOV
LYNNE O' DONNEL	AFG, AFGUAF, PPL	RODRIGO DUTERTE	PHLGOV, GOV, PHL
RODRIGO DUTERTE	PHLGOV, GOV, PHL	ANTIO CARPIO	JUD, PHL, CRM

Table 4: Spark Cluster Summary

Total Master node	1
Total slave node	10
vCPU (core) in each node	8
Memory in each node	16GB
Storage in each node	1TB

Spark-based framework with unsupervised ranking techniques with new actor aliases grouping that works on news articles collected on a periodic basis to recommend new actors. Our experimental results evaluate the framework's performance.

Currently, we limit ourselves to find new political actors but this approach can be extended to recommend new political actions in CAMEO verb dictionary and to capture the evolving roles of actors that change over time. In addition, we will extend this to build CAMEO dictionaries for other languages (e.g., Spanish, Arabic).

REFERENCES

[1] *PETRARCH.* http://petrarch.readthedocs.org/en/latest/.
[2] *practnlptools 1.0.* https://pypi.python.org/pypi/practnlptools/1.0.
[3] *Raytheon BBN Technologies.* "http://www.raytheon.com/ourcompany/bbn/".
[4] *RSS Whitelist.* https://github.com/openeventdata/scraper/blob/master/whitelist_urls.csv.
[5] Samet Atdağ and Vincent Labatut. 2013. A comparison of named entity recognition tools applied to biographical texts. In *Systems and Computer Science (ICSCS), 2013 2nd International Conference on.* IEEE, 228–233.
[6] Olga Babko-Malaya. 2005. Propbank annotation guidelines. (2005).
[7] Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1.* Association for Computational Linguistics, 86–90.
[8] John Beieler, Patrick T Brandt, Andrew Halterman, Phillip A Schrodt, and Erin M Simpson. 2016. Generating Political Event Data in Near Real Time: Opportunities and Challenges. *Computational Social Science: Discovery and Prediction. ed. by R. Michael Alvarez, Cambridge, Cambridge University Press* (2016), 98–120.
[9] Elizabeth Boschee, Premkumar Natarajan, and Ralph Weischedel. 2013. Automatic extraction of events from open source text for predictive forecasting. In *Handbook of Computational Approaches to Counterterrorism.* Springer, 51–67.
[10] Andrei Z Broder. 1997. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings.* IEEE, 21–29.
[11] CAMEO. *Conflict and Mediation Event Observations (CAMEO) Codebook.* CAMEO. "http://eventdata.parusanalytics.com/data.dir/cameo.html".
[12] George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation.
[13] Mihai Georgescu, Nattiya Kanhabua, Daniel Krause, Wolfgang Nejdl, and Stefan Siersdorfer. 2013. Extracting event-related information from article updates in wikipedia. In *European Conference on Information Retrieval.* Springer, 254–266.
[14] Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics* 28, 3 (2002), 245–288.
[15] IBM Watson. *AlchemyAPI.* IBM Watson. http://www.alchemyapi.com/.
[16] Eric Joanis and Suzanne Stevenson. 2003. A general feature space for automatic verb classification. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1.* Association for Computational Linguistics, 163–170.
[17] Jay Kreps, Neha Narkhede, Jun Rao, and others. Kafka: A distributed messaging system for log processing. In Proceedings of the NetDB (pp. 1-7).
[18] Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, Vol. 10. 707.
[19] Hao Lou, Shenghong Li, and Yuxin Zhao. 2013. Detecting community structure using label propagation with weighted coherent neighborhood propinquity. *Physica A: Statistical Mechanics and its Applications* 392, 14 (2013), 3095–3105.
[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems.* 3111–3119.
[21] MongoDB. *MongoDB.* MongoDB. "https://www.mongodb.com.
[22] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30, 1 (2007), 3–26.
[23] Open Event Data Alliance. *Web Scraper.* Open Event Data Alliance. http://oeda-scraper.readthedocs.io/en/latest.
[24] Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics* 31, 1 (2005), 71–106.
[25] Alan Ritter, Oren Etzioni, Sam Clark, and others. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 1104–1112.
[26] Kepa Joseba Rodriquez, Mike Bryant, Tobias Blanke, and Magdalena Luszczynska. 2012. Comparison of named entity recognition tools for raw OCR text.
[27] Parang Saraf and Naren Ramakrishnan. 2016. EMBERS AutoGSR: Automated Coding of Civil Unrest Events. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016.* 599–608. DOI: http://dx.doi.org/10.1145/2939672.2939737
[28] Philip A Schrodt and Deborah J Gerner. 2004. An event data analysis of third-party mediation in the Middle East and Balkans. *Journal of Conflict Resolution* 48, 3 (2004), 310–330.
[29] Philip A Schrodt and David Van Brackle. 2013. Automated coding of political event data. In *Handbook of Computational Approaches to Counterterrorism.* Springer, 23–49.
[30] Mohiuddin Solaimani, Rajeevardhan Gopalan, Latifur Khan, Patrick T Brandt, and Bhavani Thuraisingham. 2016. Spark-Based Political Event Coding. In *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService).* IEEE, 14–23.
[31] Stanford. *Stanford CoreNLP.* Stanford. "http://nlp.stanford.edu/software/corenlp.shtml".
[32] The Apache Software Foundation. *Apache OpenNLP.* The Apache Software Foundation. "http://opennlp.apache.org/".
[33] The University of Pennsylvania. *The Penn Treebank Project.* The University of Pennsylvania. "https://www.cis.upenn.edu/~treebank/".
[34] Thomson Reuters. *Open Calais.* Thomson Reuters. "http://www.opencalais.com/".
[35] Tim Weninger, Rodrigo Palacios, Valter Crescenzi, Thomas Gottron, and Paolo Merialdo. 2016. Web Content Extraction: a MetaAnalysis of its Past and Thoughts on its Future. *ACM SIGKDD Explorations Newsletter* 17, 2 (2016), 17–23.
[36] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. 2012. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. *9th USENIX conference on Networked Systems Design and Implementation* (2012), 2–2.